重视大脑的学习指南

Head First HTML5 Programming（中文版）

用JavaScript构建Web应用的初学者指南



了解HTML5高手的秘密





把HTML5和JavaScript牢牢记在你的大脑里



发现为什么你的朋友对视频的所有了解可能是错的



避免浏览器支持问题的尴尬



当心常见的浏览器陷阱

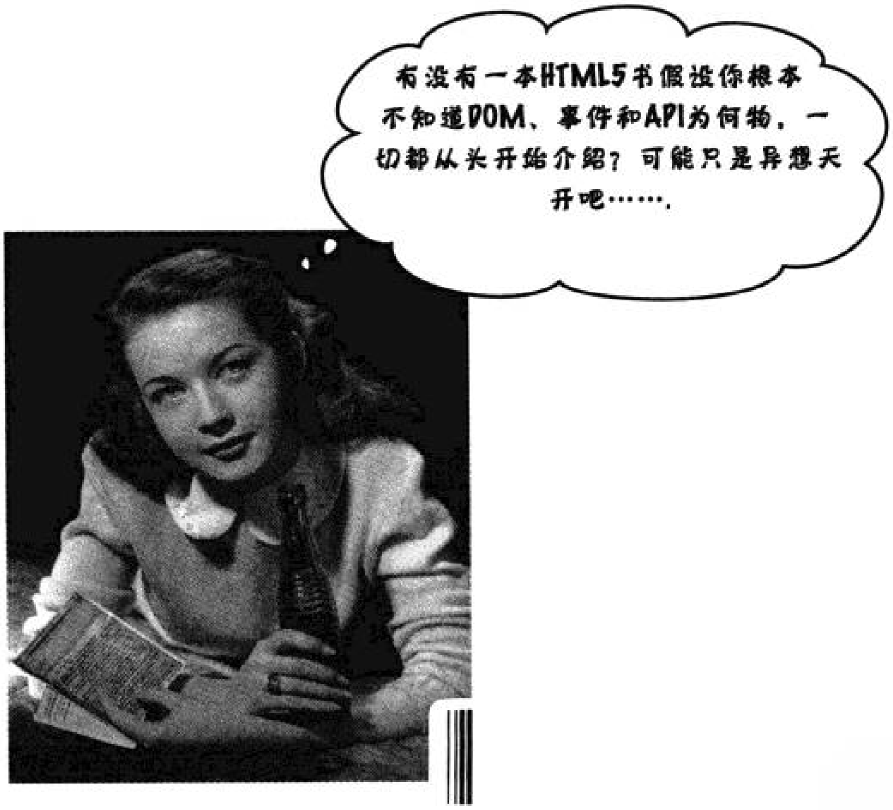
Eric Freeman & Elisabeth Robson 著

林琪 张伶 等译

O'REILLY® 中国电力出版社

Head First HTML5 Programming中文版

用Javascript构建Web应用



Eric Freeman
Elisabeth Robson 著
林琪 张伶 等译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc.授权中国电力出版社出版

中国电力出版社

图书在版编目（CIP）数据

Head First HTML5 Programming：中文版/（美）弗里曼（Freeman, E.T.），（美）罗布森（Robson, E.）著；林琪等译，—北京：中国电力出版社，2012.4

ISBN 978-7-5123-2935-5

I. ①H… II.①弗… ②罗… ③林… III. ①超文本标记语言，HTML 5－程序设计 IV. ①TP312

中国版本图书馆CIP数据核字（2012）第073514号

北京市版权局著作权合同登记

图字：01-2012-2804号

©2011 by O'Reilly Media, Inc.

Simplified Chinese Edition, jointly published by O'Reilly Media, Inc. and China Electric Power Press, 2011. Authorized translation of the English edition, 2009 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由O'Reilly Media, Inc.出版2009。

简体中文版由中国电力出版社出版，2011。英文原版的翻译得到 O'Reilly Media, Inc.的授权。此简体中文版的出版和销售得到出版权和销售权的所有者 —— O'Reilly Media, Inc.的许可。

版权所有，未得书面许可，本书的任何部分和全部不得以任何形式重制。

书　名/	Head First HTML5 Programming中文版
书　号/	ISBN 978-7-5123-2935-5
责任编辑/	刘炽
封面设计/	Karen Montgomery，Louise Barr，张健
出版发行/	中国电力出版社
地　址/	北京市东城区北京站西街19号（邮政编码100005）
印　刷/	汇鑫印务有限公司
开　本/	880毫米×1230毫米　20开本　31印张　839千字
版　次/	2012年9月第1版　2012年9月第1次印刷
印　数/	0001－3000册
定　价/	78.00元（册）

敬告读者

本书封底贴有防伪标签，刮开涂层可查询真伪

本书如有印装质量问题，我社发行部负责退换

版权专有　翻印必究

致Steve Jobs，感谢他努力推广HTML5，让HTML5如此热门，这本书应该能大卖吧……

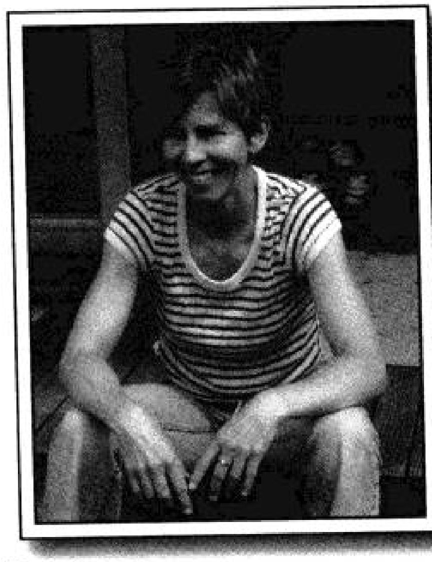
再致Steve Jobs，他是我们心目中的英雄。

《Head First HTML5 Programming》的作者



Eric Freeman

Elisabeth Robson



按照Head First系列合作者Kathy Sierra的说法，**Eric**是"少有的奇才之一，不仅语言流畅，实践经验丰富，在很多领域都表现非凡，他是hipster高手、副总裁、工程师，而且是名符其实的智多星。"

在专业领域，Eric最近刚刚离开任职近十年的一家媒体公司，他在迪斯尼公司担任Disney Online & Disney.com 的CTO。Eric现在把时间全部投入到WickedlySmart，这是他与Elisabeth共同创建的一家启动公司。

经过培训，Eric已经成为一位计算机科学家，在耶鲁大学攻读博士学位期间曾与业界杰出人物David Gelernter同窗。他的论文被认为对寻找桌面隐喻[译注]的替代品有着深远影响，这也是活动流的首个实现（活动流是他与Gelernter博士提出的一个概念）。

在闲暇时间，Eric对音乐深深着迷。在iPhone app store上可以找到Eric最近与音乐"先锋"Steve Roach合作的一个项目，名为Immersion Station。

Eric与妻子，以及小女儿居住在华盛顿州双桥岛。他女儿是Eric工作室的常客，她特别喜欢打开他的合成器和音效开关。Eric对儿童教育和营养也很关注，总在想方设法做出改善。

可以给Eric写邮件（eric@wickedlysmart.com），或者访问他的网站（http://ericfreeman.com）。

译注：桌面隐喻是在用户界面中用人们熟悉的桌面上的物品来清楚地表现计算机可处理的能力。

Elisabeth是一位软件工程师、作家和培训师。从她作为耶鲁大学学生之日起就一直热衷于技术，她在耶鲁大学获得了计算机科学硕士学位，并设计了一个并发的可视化编程语言和软件体系结构。

Elisabeth从早期就一直从事Internet的工作。她合作创建了颇有声誉的网站：The Ada Project，这是最早帮助计算机科学领域的女性在线找工作和寻求指导信息的网站之一。

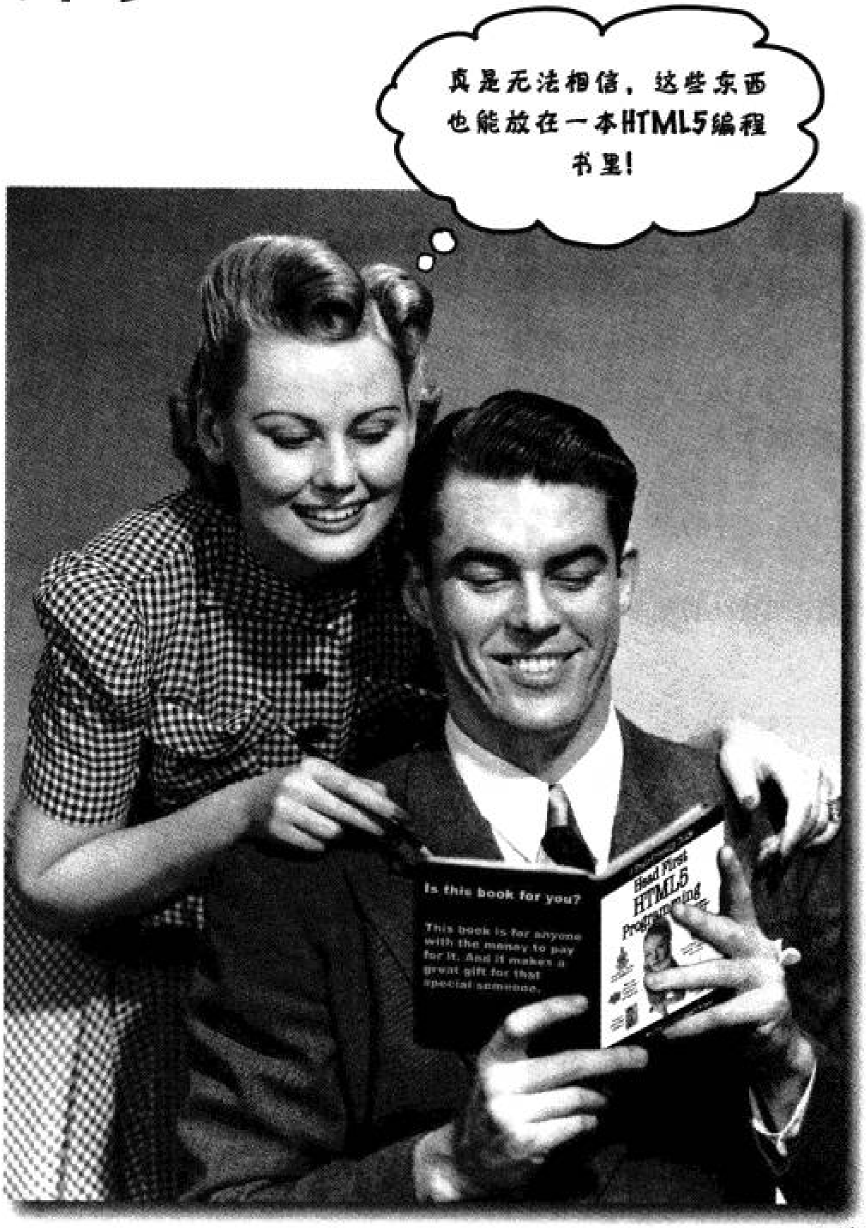
她目前是WickedlySmart的合作创始人，这是一个关注Web技术的在线教育项目，在这里她完成了有关的图书、文章、视频等。在此之前，作为 O'Reilly Media的特殊项目主任（Director of Special Projects），Elisabeth曾经在各种技术专题发布过个人研讨文档和在线课程，这使她对创建学习体验来帮助人们理解技术越来越充满热情。在O'Reilly工作之前，Elisabeth主要在迪斯尼公司播洒她的仙女魔法粉，在那里，她带领数字媒体研发力量开展研究工作。

如果没有坐在计算机前，你会发现Elisabeth总是带着她的相机在户外踏青、骑单车或者划皮艇，也可能在烹调素食大餐。

可以给她发Email（beth@wickedlysmart.com）或者访问她的博客（http://elisabethrobson.com）。

如何使用这本书

引子



有个问题真是听得我们耳朵都磨出茧了：“你们到底为什么要把这样一些东西放在一本HTML5书里呢？”。现在就来回答这个问题。

谁适合看这本书？

如果对下面的所有问题都能肯定地回答“是”：

1. 你有一台安装了Web浏览器和测试编辑器的计算机吗？

2. 你是不是想**学习**、**理解**、**记住**并且用最好的技术和最新的标准来**创建**Web应用？

3. 你是不是更喜欢一种轻松的氛围，**就像在晚餐餐桌上交谈一样，而不愿意被动地听枯燥乏味的技术报告**？

谁可能不适合看这本书？

如果满足下面任何一种情况：

1. 你是不是对于写Web页面一无所知？

可以参考《Head First HTML with CSS & XHTML》，这本书很好地介绍了Web开发，然后再回来加入我们的HTML5之旅。

2. 你是不是在开发Web应用，正在找一本关于HTML5的参考书？

3. 你是不是对**新鲜事物都畏首畏尾**？只喜欢简单的样式，而不敢尝试把条纹和格子混在一起看看？你是不是觉得，如果加入了cheesy 50的教育片和拟人化的JavaScript API，这样一本书肯定不是一本正经八百的技术书？

那么，这本书将不适合你。



[来自市场的声音：只要买得起，不论是用信用卡还是现金，都可以拥有这本书。现金也不错。——小编]

我们知道你在想什么。

“这算一本正经八百的HTML5编程书吗？”

“这些图做什么用？”

“我真能这样学吗？”

我们也知道你的大脑正在想什么。



你的大脑总是渴求一些新奇的东西。它一直在搜寻、审视、期待着不寻常的事情发生。大脑的构造就是如此，正是这一点才让我们不至于墨守成规，能够与时俱进。

我们每天都会遇到许多按部就班的事情，这些事情很普通，对于这样一些例行的事情或者平常的东西，你的大脑又是怎么处理的呢？它的做法很简单，就是不让这些平常的东西妨碍大脑真正的工作。那么什么是大脑真正的工作呢？这就是记住那些确实重要的事情。它不会费心地去记乏味的东西。就好像大脑里有一个筛子，这个筛子会筛掉“显然不重要”的东西，如果遇到的事情枯燥乏味，这些东西就无法通过这个筛子。

那么你的大脑怎么知道到底哪些东西重要呢？打个比方，假如你某一天外出旅行，突然一只大老虎跳到你面前，此时此刻，你的大脑和身体会做何反应？

神经元会“点火”，情绪爆发，释放出一些化学物质。

好了，这样你的大脑就会知道……

这肯定很重要！可不能忘记了！



不过，假如你正待在家里或者坐在图书馆里，这里很安全、很舒适，肯定没有老虎。你正在刻苦学习，准备应付考试。也可能想学一些比较难的技术，你的老板认为掌握这种技术需要一周时间，最多不超过十天。

这就存在一个问题。你的大脑很想给你帮忙。它会努力地把这些显然不太重要的内容赶走，保证这些东西不去侵占本不算充足的脑力资源。这些资源最好还是用来记住那些确实重要的事情，比如大老虎，再比如遭遇火灾等。就像你的大脑会让你记住绝对不能再穿着短裤去滑雪。

没有一种简单的办法来告诉大脑：“嘿，大脑，真是谢谢你了，不过不管这本书多没意思，也不管现在我对它多么无动于衷，但我确实希望你能把这些东西记下来。”

我们认为"Head First"的读者就是要学习的人。

那么，怎么学习呢？首先必须获得知识，然后保证自己确实不会忘记。这可不是填鸭式的硬塞。根据认知科学、神经生物学和教育心理学的最新研究结果，学习的途径相当丰富，绝非只是通过书本上的文字。我们很清楚怎么让你的大脑兴奋起来。

下面是一些Head First学习原则：



看得到。与单纯的文字相比，图片更能让人记得住，通过图片，学习效率会更高（对于记忆和传递型的学习，甚至能有多达89%的效率提升）。而且图片更能让人看懂。以往总是把图片放在一页的最下面，甚至放在另外的一页上，与此不同，如果把文字放在与之相关的图片内部，或者在图片的周围写上相关文字，学习者的能力就能得到多至两倍的提高，从而能更好地解决有关的问题。

采用一种针对个人的交谈式风格。最新的研究表明，如果学习过程中采用一种第一人称的交谈方式直接向读者讲述有关内容，而不是用一种干巴巴的语调介绍，学生在学习之后的考试中成绩会提高40%。正确的做法是讲故事，而不是做报告。要用通俗的语言。另外不要太严肃。如果你面对着这样两个人，一个是你在餐会上结识的很有意思的朋友，另一个人学究气十足，喋喋不休地对你说教，在这两个人中，你会更注意哪一个呢？

让学习的人想得更深。换句话说，除非你很积极地让神经元活动起来，否则你的头脑里什么也不会发生。必须引起读者的好奇，促进、要求并鼓励读者去解决问题、得出结论、产生新的知识。为此，需要发出挑战，留下练习题和拓宽思路的问题，并要求读者完成一些实践活动，让左、右脑都开动起来，而且要利用到多种思维。



引起读者的注意，而且要让他一直保持注意。我们可能都有过这样的体验，"我真的想把这个学会，不过看过一页后实在是让我昏昏欲睡"。你的大脑注意的是那些不一般、有意思、有些奇怪、抢眼的、意料之外的东西。学习一项有难度的新技术并不一定枯燥。如果学习过程不乏味，你的大脑很快就能学会。



影响读者的情绪。现在我们知道了，记忆能力很大程度上取决于所记的内容对我们的情绪有怎样的影响。如果是你关心的东西，就肯定记得住。如果让你感受到了什么，这些东西就会留在你的脑海中。不过，我们所说的可不是什么关于男孩与狗的伤心故事。这里所说的情绪是惊讶、好奇、觉得有趣、想知道"什么……"，还有就是一种自豪感，如果你解决了一个难题，学会了所有人都觉得很难的东西，或者发现你了解的一些知识竟是那些自以为无所不能的傲慢家伙所不知道的，此时就会有一种自豪感油然而生。

元认知：有关思考的思考

如果你真的想学，而且想学得更快、更深，就应该注意你怎样才会专注起来，考虑自己是怎样思考的，并了解你的学习方法。

我们中间大多数人长这么大可能都没有上过有关元认知或学习理论的课程。我们想学习，但是很少有人教我们怎么来学习。

不过，这里可以做一个假设，如果你手上有这本书，你想学习项目管理，而且可能不想花太多时间。如果你想把这本书中读到的知识真正用起来，就需要记住你读到的所有内容。为此，必须理解这些内容。要想最大程度地利用这本书或其他任何一本书，或者掌握学习经验，就要让你的大脑负起责来，要求它记住这些内容。



怎么做到呢？技巧就在于要让你的大脑认为你学习的新东西确实很重要，对你的生活有很大影响。就像老虎出现在面前一样。如若不然，你将陷入旷日持久的拉锯战中，虽然你很想记住所学的新内容，但是你的大脑却会竭尽全力地把它们拒之门外。

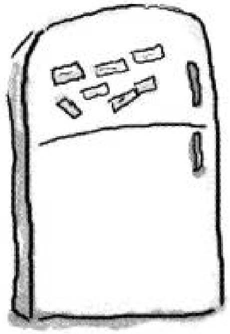
那么究竟怎样才能让大脑把HTML5（和JavaScript）开发看做是一只饥饿的老虎呢？

这有两条路，一条比较慢，很乏味。另一条路不仅更快，还更有效。慢方法就是大量地重复。你肯定知道，如果反反复复地看到同一个东西，即便再没有意思，你也能学会并记住。如果做了足够的重复，你的大脑就会说，“尽管看上去这对他来说好像不重要，不过，既然他这样一而再、再而三地看同一个东西，所以我假定这应该是重要的。”

更快的方法是**尽一切可能让大脑活动起来**，特别是开动大脑来完成不同类型的活动。如何做到这一点呢？上一页列出的学习原则正是一些主要的可取做法，而且经证实，它们确实有助于让你的大脑全力以赴。例如，研究表明，把文字放在所描述图片的中间（而不是放在这一页的别处，比如作为标题，或者放在正文中），这样会让你的大脑更多地考虑这些文字与图片之间有什么关系，而这就会让更多的神经元点火。让更多的神经元点火 = 你的大脑更有可能认为这些内容值得关注，而且很可能需要记下来。

交谈式风格也很有帮助，当人们意识到自己在与“别人”交谈时，往往会更专心，这是因为他们总想跟上谈话的思路，并能做出适当的发言。让人惊奇的是，大脑并不关心“交谈”的对像究竟是谁，即使你只是与一本书“交谈”，它也不会在乎！另一方面，如果写作风格很正统、干巴巴的，你的大脑就会觉得，这就像坐在一群人当中被动地听人做报告一样，很没意思，所以不必在意对方说的是什么，甚至可以打瞌睡。

不过，图片和交谈风格还只是开始而已，能做的还有很多。



可以用下面的方法让你的大脑就范

好了，我们该做的已经做了，剩下的就要看你自己的了。以下提示可以作为一个起点：听一听你的大脑是怎么说的，弄清楚对你来说哪些做法可行，哪些做法不能奏效。要尝试新鲜事物。

把这一页撕下来，贴到你的冰箱上。

---

① 慢一点。你理解得越多，需要记的就越少。

不要光是看看就行了。停下来，好好想一想。书中提出问题的时候，你不要直接去翻答案。可以假想真的有人在问你这个问题。你让大脑想得越深入，就越有可能学会并记住它。

② 做练习，自己记笔记。

我们留了练习，但是如果这些练习的解答也由我们一手包办，那和有人替你参加考试有什么分别？不要只是坐在那里看着练习发呆。拿出笔来，写一写画一画。大量研究都证实，学习过程中如果能实际动动手，这将改善你的学习。

③ 阅读“没有傻问题”。

顾名思义。这些问题不是可有可无的旁注，它们绝对是核心内容的一部分！千万不要跳过去不看。

④ 上床睡觉之前不要再看别的书，至少不要看其他有难度的东西。

学习中有一部分是在你合上书之后完成的（特别是，要把学到的知识长久地记住，这往往无法在看书的过程中做到）。你的大脑也需要有自己的时间，这样才能再做一些处理。如果在这段处理时间内你又往大脑里灌输了新的知识，那么你刚才学的一些东西就会丢掉。

⑤ 要喝水，而且要多喝点水。

能提供充足的液体，你的大脑才能有最佳表现。如果缺水（可能在你感觉到口渴之前就已经缺水了），学习能力就会下降。

⑥ 讲出来，而且要大声讲出来。

说话可以刺激大脑的另一部分。如果你想看懂什么，或者想更牢地记住它，就要大声地说出来。更好的办法是，大声地解释给别人听。这样你会学得更快，而且可能会有以前光看不说时不曾有的新发现。

⑦ 听听你的大脑怎么说。

注意一下你的大脑是不是负荷太重了。如果发现自己开始浮光掠影地翻看，或者刚看的东西就忘记了，这说明你该休息一会了。达到某个临界点时，如果还是一味地向大脑里塞，这对于加快学习速度根本没有帮助，甚至还可能影响正常的学习进程。

⑧ 要有点感觉。

你的大脑需要知道这是很重要的东西。要真正融入到书中的故事里。为书里的照片加上你自己的图题。你可能觉得一个笑话很憋脚，不太让人满意，但这总比根本无动于衷要好。

⑨ 真正做些工作!

把这些知识应用到你的日常工作中去；利用你所学的想法完成项目决策。具体做些工作，你会得到书中练习和活动以外的经验。所需要的只是一支笔和一个要解决的问题……通过使用针对考试所学的工具和技术可以更好地解决这个问题。

重要说明

要把这看做是一个学习过程，而不要简单地把它看成是一本参考书。我们在安排内容的时候有意做了一些删减，只要是对有关内容的学习有妨碍，都毫不留情地把这些部分一律删掉。另外，第一次看这本书的时候，要从第一页从头看起，因为书中后面的部分会假定你已经看过而且学会了前面的内容。

希望你了解HTML和CSS。

如果你不懂HTML标记（也就是说，关于HTML文档的所有内容，包括元素、属性、属性结构、结构与表现），读这本书之前请先找一本《Head First HTML with CSS & XHTML》看看。如果你对HTML已经有所了解，应该能顺利地读下去。

如果对JavaScript有一些经验会有帮助，不过不要求你了解JavaScript。

如果你有编程或编写脚本（甚至不是JavaScript）的技术背景，肯定会有帮助。不过，读这本书并不要求你懂JavaScript。实际上，这本书设计作为《Head First HTML with CSS & XHTML》的续篇，而那本书中并没有脚本。

建议你学习这本书时最好使用多种浏览器。

建议你用不同的浏览器测试这本书中的页面和Web应用。这会让你了解不同浏览器之间的差别，另外也能对创建适用于多种浏览器的页面有所认识。我们强烈推荐学习这本书时使用Google Chrome和Apple Safari，因为一般来讲，它们最符合当前的标准。不过我们真的建议你还要尝试其他主流浏览器的当前版本，包括Internet Explorer、Firefox和Opera，以及安装有iOS和Android系统的设备上的移动浏览器。

书里的实践活动绝不是可有可无。

这里的练习和实践活动不是可有可无的装饰和摆设，它们也是这本书核心内容的一部分。其中有些练习和活动有助于记忆，有些能够帮助你理解，还有一些对于如何应用所学的知识很有帮助。千万不要把这些练习跳过不做。甚至填字游戏也很重要，它们可以帮助你将有关概念植入你的大脑。不过更重要的是，它们可以让你的大脑有机会在不同的上下文考虑这些词汇和概念。

我们有意安排了许多重复，这些重复非常重要。

Head First系列图书有一个与众不同的地方，这就是，我们希望你确确实实地掌握这些知识，另外希望在学完这本书之后你能记住学过了什么。大多数参考书都不太重视重复和回顾，但是由于这是一本有关学习的书，你会看到一些概念一而再、再而三地出现。

“Brain Power（头脑风暴）”练习没有答案。

有一些头脑风暴练习根本没有正确的答案，另外一些练习中，头脑风暴实践活动的一部分学习过程就是让你确定你的答案是否正确以及在何种情况下正确。有些头脑风暴练习中，你会得到一些提示，为你指明正确的方向。

软件需求

要编写HTML5和JavaScript代码，你需要有一个文本编辑器、一个浏览器，有时还需要一个Web服务器（可以本地安装在你的个人台式机上）。

对于Windows，我们推荐的文本编辑器是PSPad、TextPad或EditPlus（不过，如果别无选择，也可以使用Notepad）。对于Mac我们推荐的文本编辑器包括TextWrangler、TextMate或TextEdit。如果你使用的是一个Linux系统，其中已经内置了大量文本编辑器，相信不需要我们多说了。

希望你已经了解浏览器，而且至少安装了两个浏览器（见上一页的说明）。如果还没有，现在就动手安装吧。另外学习如何使用浏览器开发工具也很有意义。每个主流浏览器都有一些内置工具，可以用来检查JavaScript控制台（使用console.log显示时，除了输出之外还能看到错误输出，这个工具很方便，可以作为`alert`的候选）、Web存储使用、DOM、应用到元素的CSS样式，诸如此类。有些浏览器甚至还有插件来提供另外的开发工具。学习这本书并不需要这些开发工具，不过如果你想花些时间来研究如何使用这些工具，这会让你的开发更为容易。

有些HTML5特性和JavaScript API要求从一个实际Web服务器提供文件而不是通过加载文件（也就是说，你的URL要以http://开头，而不是`file://`）。我们会在这本书中适当的地方明确指出哪些例子需要有一个服务器，不过，如果你愿意，建议现在就在你的计算机上安装一个服务器。对于Mac和Linux，已经内置了Apache，所以只需要知道如何访问这个服务器以及把文件放在哪里，从而能够使用你的本地服务器提供这些文件。对于Windows，则需要安装Apache或IIS；如果你选择安装Apache，现在已经有丰富的开源工具（如WAMP和XAMPP），安装相当容易。

该说的都已经说了！祝你开心……

技术审校团队

Paul Barry



他可不只是个普通的审校人员，Paul是一位有丰富经验的"Head First"作者，著有《Head First Python》和《Head First Programming》。

David Powers



我们的技术主审。

Bert Bates



这里的审校人员都很不一般，他也是这个系列的作者！嘿，真有压力啊……

Lou Barr



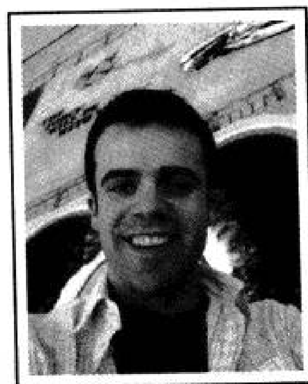
我们原来想告诉她只需要帮我们完成图片部分，结果她实在忍不住，也成为我们的一位重要的技术审校。

Rebeca Duhn-Kahn



Rebeca就像我们的第二双眼睛，她帮我们检查别人都看不出来的代码细节（包括我们自己）！

Trevor Farlow



最尽心的技术审校，他的投入算得上110%。他甚至半夜穿着睡衣四处游荡来测试我们的地理定位代码。

致我们的技术审校人员：

非常感谢我们的技术审校团队。这个团队再次向我们证明了我们是多少需要他们的技术经验和对细节问题的关注。David Powers、Rebeca Dunn-Krahn、Trevor Farlow、Paul Barry、Louise Barr和Bert Bates在审校时都不遗余力，全凭大家的努力，这本书比原先好太多了。你们太棒了！

致谢

致更多技术审校：

我们的书中总会有他，不过这里还是想对David Powers再说两句。David Powers是我们最尊敬的技术审校，他也是很多书的作者，包括《PHP Solutions: Dynamic Web Development Made Easy》。David的建议总能让我们的文字大有起色，如果经过了David的审查，我们就会很放心，晚上能睡得更好，也能达到我们的技术目标。David，再次感谢！

致编辑：我们后面3本书能不能锁定他啊？能不能作为我们的专用审校？

致O'Reilly人员：

Courtney Nash肩负着最艰难的任务管理工作，不仅仅是这本《Head First HTML5 Programming》，嗯，她还要管理我们。Courtney不仅为我们清空了道路上的所有障碍，同时就像每一位编辑需要做的一样，还要向我们施加必要的压力，终于让这本书得以问世。不过，最重要的是，Courtney还对这本书及内容提供了非常有价值的反馈，使这本书有了几次重要的返工。正是由于Courtney的努力，让这本书更为出色。谢谢你。



Courtney Nash



还是Lou Barr!

Lou Barr也是这本书不可或缺的一部分，对于这本书，她身兼很多角色。不仅是技术审校人员、平面设计人员、Web设计人员，同时也是Photoshop处理人员。谢谢你，Lou，没有你我们绝无可能完成！

还要感谢帮助过我们的其他人员：

还要感谢O'Reilly的其他人员，感谢大家各种各样不同方式的支持。这个团队包括Mike Hendrickson、Mike Loukides、Laurel Ruma、Karen Shaner、Sanders Kleinfeld、Kristen Borg、Karen Montgomery、Rachel Monaghan、Julie Hawks和Nancy Reinhardt。

更多致谢！*

还要感谢另外的很多人：

James Henstridge编写的代码后来成为第10章中的分形浏览器，我们根据书中的用途做了调整，以便在这本书中使用。我们的代码可能没有他原先那个版本精巧，对此我们深表歉意。Laurence Zankowski是一位演员，同时也是一位艺术家，在这本书中再次以Starbuzz CEO的形象出现，还帮我们测试了第8章中的视频应用（务必要看）。Bainbridge Island Downtown Association慷慨地允许我们使用他们绝妙的logo（由Denise Harris设计）作为WickedlySmart总部的logo。感谢Anthony Vizzari和A&A Studios允许我们使用他们非凡的摄像间照片。



他又又又……回来了！

我们的TweetShirt启动项目示例使用了ChethStudios.Net的一些好看的图标。非常感谢Internet Archive的辛勤工作，我们在Webville TV中用到的电影就来源于此。还要感谢Daniel Steinberg总能为我们提出建议。

最后要感谢Kathy和Bert。

最后，但绝不是不重要，要感谢Kathy Sierra和Bert Bates，我们最重要的合作者,也是创造这个系列的核心人物。再一次，希望我们没有为这个系列丢脸。

Bert Bates

Kathy Sierra



正在忙着完成《Head First Parelli》。

*之所以要感谢这么多人，这是因为我们测试过，书中致谢提到的每一个人都起码会买上一本，如果亲戚朋友多，可能还会买更多本。如果你希望在我们下本书的致谢中出现，而且你有一个大家族，请联系我们。

Safari®图书在线

Safari图书在线是一个按需提供资源的数字图书馆，从中可以很容易地搜索7500本技术书、参考书和视频，快速找到你想要的答案。

只需订阅，就可以从我们的在线图书馆阅读任何页面，观看任何视频。你可以在你的手机和移动设备上看书，能够在新书出版之前获得书目，还可以享有特权查看正在编写的书稿并向作者提出反馈意见。你可以剪切粘贴代码示例、整理最喜欢的图书、下载你需要的章节、为重要内容设置书签、创建笔记、打印页面，此外还有大量特性可以节省时间，让你从中受益。

O'Reilly Media已经将这本书（英文版）上传到Safari图书在线服务。要想通过数字方式全面访问这本书以及O'Reilly和其他出版商提供的其他类似图书，可以免费注册 *http://my.safaribooksonline.com*。

目录（概览）

	引子	ix
1	认识HTML5：欢迎来到Web镇	1
2	介绍JavaScript和DOM：一点点代码	35
3	事件、处理程序，诸如此类：一点点交互	85
4	JavaScript 函数和对象：正式JavaScript	113
5	实现HTML位置感知：地理定位	165
6	与Web交流：喜欢社交的应用	213
7	秀出你的艺术天份：画布	281
8	不再是父辈的老电视：视频……以及特邀演员“画布”	349
9	在本地存储：Web存储	413
10	运用javascript：Web工作线程	473
	附录：（我们没有谈到的）十大主题	531
	索引	549

详细目录

引子

你的大脑与HTML编程。你想学些新东西，但是你的大脑总是帮倒忙，它会努力让你记不住所学的东西。你的大脑在想：“最好留出空间来记住那些确实重要的事情，比如要避开哪个野生动物，还有裸体滑雪是不是不太好。”那么，如何让你的大脑就范？让它认为如果不知道HTML5和JavaScript你将无法生存！

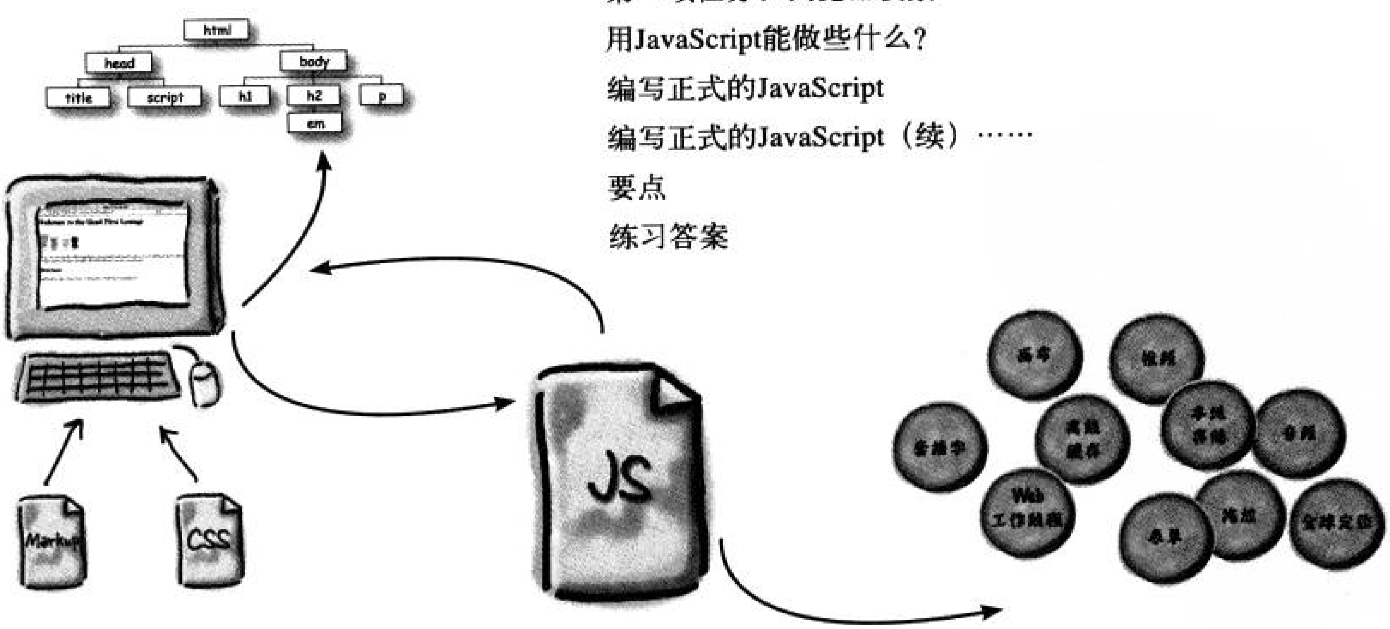
谁适合看这本书？	xxii
我们知道你在想什么	xxiii
我们也知道你的大脑正在想什么	xxiii
元认知：有关思考的思考	xxv
技术审校团队	xxx
致谢	xxxi

1 认识HTML5

欢迎来到Web镇

HTML正发展得如火如荼。确实，起初HTML只是一个标记语言，不过最近HTML早已有了新的功用。如今我们得到的这个语言已经特别为构建真正的Web应用做了调整，提供了本地存储、2D绘图、离线支持、套接字和线程等诸多特性。HTML的发展当然不会是一帆风顺，其中充满了戏剧性的变故（这些我们都将一一介绍），不过，这一章中，我们首先会去Web镇兜兜风，感受一下“HTML5”都有些什么。来吧，上车，向Web镇前进，让我们从零开始认识HTML5。

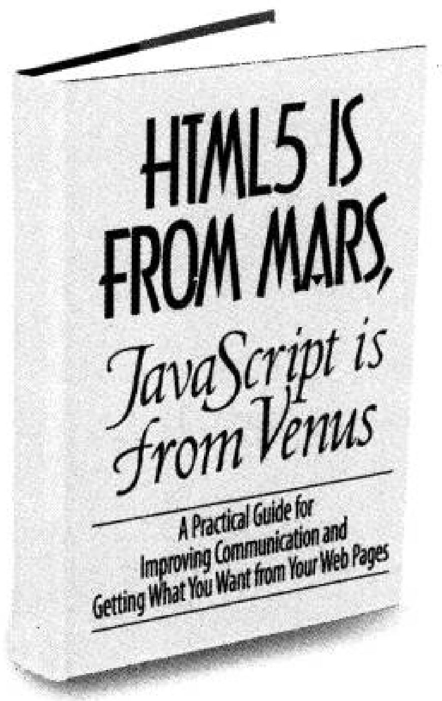
今天就升级到HTML5!	2
HTML5-o-Matic介绍，着手更新你的HTML!	4
没想到吧，HTML5标记居然离你这么近!	7
HTML5闪亮登场：最新版HTML的告白	11
真正的HTML5请起立……	12
HTML5到底如何工作……	14
谁做什么?	16
第一项任务：浏览器侦察	17
用JavaScript能做些什么?	22
编写正式的JavaScript	25
编写正式的JavaScript（续）……	26
要点	31
练习答案	33



2 介绍JavaScript和DOM

一点点代码

JavaScript会带你进入新境界。你已经了解了HTML标记（也称为结构），而且知道了CSS样式（也叫做表示），剩下的就是JavaScript（这也称为行为）。如果你只知道结构和表示，当然了，创建一个漂亮的页面是没有问题的，不过它们只是页面而已。用JavaScript增加行为时，你就能创建一种交互式的体验。或者，还可以更棒，你能创建完备的Web应用。准备好了吗？下面就在你的Web工具箱里增加一项最有意思、功能最全的技能：JavaScript和编程！



JavaScript的工作方式	36
用JavaScript能做什么？	37
声明变量	38
如何命名变量	40
需要表达	43
反反复复……	46
用JavaScript做判断	49
更多判断…… 另外，增加一个“收容箱”	50
在页面中增加JavaScript，怎么加？在哪里加？	53
JavaScript如何与页面交互	54
如何制作你自己的DOM	55
初尝DOM	56
HTML5来自火星，JavaScript来自金星	58
页面完全加载之前不要打扰DOM	64
那么，DOM还能做什么？	66
能不能再谈谈JavaScript？ 或者，能不能告诉我JavaScript中如何存储多个值？	67
Phrase-o-Matic	71
要点	75
练习答案	77

3 事件、处理程序，诸如此类

一点点交互

你还没有与用户接触呢。你已经了解JavaScript的基础知识，不过你能与你的用户交互吗？如果页面能响应用户的输入，它们就不再只是文档了，而是有生命的、有反应的应用。在这一章中，你将学习如何处理一种用户输入形式（嘿嘿，这是个双关语），并把老式的HTML <form>元素关联到具体的代码。听上去有些危险，不过这样确实功能强大。系上安全带，这一章速度会很快，我们将从零出发，转瞬间就会到达交互式应用。



准备进入Web镇之声	86
出发……	87
不过我点击“Add Song”按钮时什么也没有发生	88
处理事件	89
制订计划……	90
访问“Add Song”按钮	90
为按钮指定一个点击处理程序	91
仔细研究发生了什么……	92
获得歌曲名	94
如何向页面增加一首歌？	97
如何创建一个新元素	99
向DOM增加新元素	100
集成在一起……	101
……试一试	101
回顾—我们做的工作	102
如何增加成品代码……	105
集成成品代码	106
要点	108
练习答案	110

4 JavaScript函数和对象

正式JavaScript

你能把自己称为脚本开发人员吗？ 可能吧，你已经懂得不少JavaScript了。不过，既然能作为一个程序员，谁还只想当个区区脚本开发人员呢？现在严肃一点，是时候了，你该学习学习函数和对象。要想编写更强大、更有组织，而且更易于维护的代码，它们正是关键所在。另外，HTML5 JavaScript API中也大量使用了函数和对象，所以你对它们越了解，就能越快地熟悉新API并熟练掌控。系上安全带，这一章要求你必须全力以赴……



扩展你的词汇	114
如何增加你自己的函数	115
函数如何工作	116
函数剖析	121
局部变量和全局变量	123
了解局部变量和全局变量的作用域	124
噢，我们提到过函数也是值吗？	128
有人谈到“对象”？	131
如何用JavaScript创建对象	132
可以用对象做的一些事情	133
谈谈向函数传入对象	136
对象也可以有行为……	142
再回到Web镇影院……	143
增加“this”关键字	145
如何创建构造函数	147
到底是怎么回事？	149
试一试你的构造函数	153
window对象到底是什么？	155
再谈window.onload	156
再谈 document对象	157
再谈document.getElementById	157
再来考虑一个对象：元素对象	158
要点	160
练习答案	162

5 实现HTML位置感知

地理定位

无处可逃。有时，如果能知道你在什么位置，结果会大不相同（特别是对于一个Web应用）。这一章我们会告诉你如何创建位置感知的Web页面，有时你能精准地指出用户正站在哪个角落，有时只能确定他们位于哪个城区（不过总能知道在哪个城市）。当然了，有时你根本无法确定他们的位置，这可能是技术上的原因，也可能只是因为他们不希望你多管闲事。可以想想看。不管怎样，这一章我们开始研究一个JavaScript API：地理定位。带着你最好的位置感知设备（甚至可以是你的桌面PC），出发吧。



位置，位置，位置 166
纬度和经度…… 167
地理定位API如何确定你的位置 168
你到底在哪里？ 172
如何集成 176
找出我们的秘密位置…… 179
编写代码查找距离 181
如何向页面增加地图 183
加一个按钉…… 186
用Google Maps API还能做另外一些很酷的事情 188
可以谈谈你的精度吗？ 191
“无处可逃” 192
启动应用 193
调整原来的代码…… 194
动起来！ 196
你有一些选项…… 198
超时和最大年龄…… 199
~~不要~~大胆尝试（让地理定位充分施展） 202
完成这个应用！ 204
集成我们的新函数 205
要点 207
练习答案 210

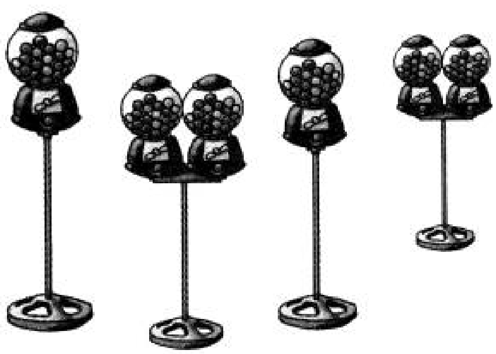
6 与Web交流

喜欢社交的应用

你留在页面里太久了。 该出去走走，与Web服务聊聊，收集些数据带回来，这样就能构建更好的应用，把所有这些数据融合起来。这是编写现代HTML5应用很重要的一部分，不过，要做到这样，你必须知道如何与Web服务交流。这一章我们就来讨论这个内容，在你的页面中结合一个实际Web服务的数据。了解如何做到之后，你就能走出去，与你想要的任何Web服务交往了。我们甚至会告诉你同Web服务交流时要用的最流行的“行话”。所以，跟我来，你会用到更多的API：通信API。



当心这一章的惊险情节！



万能糖果公司需要一个Web应用 214
万能糖果公司的更多背景介绍 216
如何向Web服务做出请求？ 219
如何从JavaScript做出请求 220
XML让位，JSON登场 226
编写一个onload处理函数 229
显示糖果销售数据 230
如何建立你自己的Web服务器 231
调整代码以利用JSON 236
转向实际服务器 237
惊险情节！ 239
记得吧？我们有一个惊险情节，这是一个bug 242
浏览器安全策略是什么？ 244
那么，我们有哪些选择？ 247
认识JSONP 252
那么JSONP中的“P”代表什么？ 253
更新万能糖果公司的Web应用 256
第1步：处理script元素…… 264
第2步：现在来建立定时器 265
第3步：重新实现JSONP 267
差点忘了：当心可怕的浏览器缓存 272
如何删除重复的销售报告 273
更新JSON URL来包含lastreporttime 275
要点 277

8 不再是父辈的老电视

视频……以及特邀演员“画布”

不再需要插件。毕竟，视频现在是HTML家族的直系成员。把<video>元素直接放在页面中，你就能立即看到视频，而且大多数设备上都能支持。不过，视频绝不仅仅是一个元素，它还是一个JavaScript API，有了它，我们可以控制视频的播放、创建自己的定制视频界面，还可以用全新的方式集成视频与HTML的其余部分。说到集成……还记得我们一直在谈论的视频与画布的联系吧。你会看到，视频和画布在一起会为我们提供一种强有力的新方法来实时地处理视频。这一章我们先在页面中玩转视频，然后再来研究JavaScript API。来吧，只用一点点标记、JavaScript、视频与画布，结果会让你瞠目结舌。

认识Webville TV 350
先搞定HTML部分…… 351
video元素如何工作？ 353
深入研究视频属性…… 354
关于视频格式需要知道什么 356
如何处理所有这些格式…… 358
我听说有API 363
Webville TV的一个小内容“计划” 364
如何编写“视频结束”处理程序 367
canPlayType方法如何工作 369
打开演示样机的包装 375
检查其余工厂代码 376
setEffect和setVideo处理程序 378
实现视频控件 384
切换测试视频 387
现在来些特效 389
如何完成视频处理 392
如何使用scratch缓冲区处理视频 393
用画布实现scratch缓冲区 395
现在需要写一些效果 399
如何使用error事件 406
要点 408
练习答案 410



打开Webville TV……

10 运用JavaScript

Web工作线程

脚本运行缓慢，你还想继续运行吗？ 如果你使用JavaScript或浏览Web的时间足够长，可能见过“slow　script”消息告诉你脚本运行缓慢。不过，既然你的新机器里已经拥有那些多核处理器，脚本怎么可能会运行得慢呢？这是因为，JavaScript一次只能做一件事。不过，有了HTML5和Web工作线程，一切都改变了。现在完全可以创建你自己的JavaScript工作线程来完成更多工作。也许你只是想设计一个更有响应性的应用，或者可能只想最大限度地发挥机器的CPU能力，不论怎样，Web工作线程都竭诚为你提供帮助。带上你的JavaScript经理帽，找些工人干活吧！

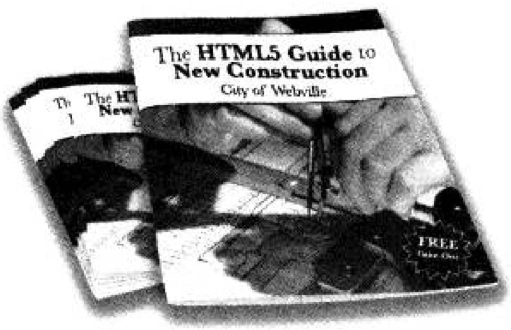


可怕的“slow script”	474
JavaScript如何分配时间	474
单线程遇到麻烦	475
增加另一个控制线程提供帮助	476
Web工作线程如何工作	478
第一个Web工作线程……	483
编写Manager.js	484
从工作线程接收消息	485
现在来编写工作线程	486
虚拟土地掠夺	494
如何计算Mandelbrot集	496
如何使用多个工作线程	497
构建Fractal Explorer应用	503
成品代码	504
创建工作线程，为它们分配任务……	508
编写代码……	509
启动工作线程	510
实现工作线程	511
重回代码：如何处理工作线程的结果	514
让画布占满浏览器窗口	517
吹毛求疵的主程序员	518
实验室生活	520
要点	524
练习答案	526

附录：其他

我们已经介绍了很多，这本书也快要结束了。我们会想你的，不过在你离开之前，还要再做一点准备，否则我们实在不放心让你贸然进入这个纷乱的世界。我们不可能把你需要知道的一切都放在这样短短的一章里。实际上，我们原先确实加入了（其他章节中没有谈到的）需要了解的有关HTML5的所有内容，只不过把字体缩小到0.00004。这样才能放得下，可惜没有人能看得清。所以，我们最后还是删去了大部分内容，只把最重要的部分保留在这个“十大主题”附录中。





#1 Modernizr 532

#2 音频 533

#3 jQuery 534

#4 XHTML死了，还是XHTML永存 536

#5 SVG 537

#6 离线Web应用 538

#7 Web Socket 539

#8 更多画布API 540

#9 选择器API 542

#10 不过，还有呢！ 543

HTML5新构造指南 545

Web镇HTML5语义元素指南 546

Web镇CSS3属性指南 548

i

索引 549

pin51.com 拼吾爱 最新资源分享下载站

1 认识HTML5

欢迎来到Web镇



HTML正发展得如火如荼。确实，起初HTML只是一个标记语言，不过最近HTML早已有了新的功用。如今我们得到的这个语言已经特别为构建真正的Web应用做了调整，提供了本地存储、2D绘图、离线支持、套接字和线程等诸多特性。HTML的发展当然不会是一帆风顺，其中充满了戏剧性的变故（这些我们都将一一介绍），不过，这一章中，我们首先会去Web镇兜兜风，感受一下“HTML5”都有些什么。来吧，上车，向Web镇前进，让我们从零开始认识HTML5。

注意：XHTML在2009年收到一封抬头是“Dear John”的信，在后面我们将在“它们现在的发展”部分介绍XHTML。

升级到HTML5，就在今天！还等什么？
快来试用我的
HTML5-o-Matic
只需要3个简单步骤就能搞定。



快行动吧！ 只用一点点时间，外加3个简单步骤，就能把你的蹩脚的老HTML页面升级到HTML5。

真的这么容易吗？
绝对没问题.实际上我们已经为你准备了一个演示。

先来看看下面这个年代久远的陈旧、过时的HTML；
我们要在你面前让它摇身一变成为HTML5：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Head First Lounge</title>
    <link type="text/css" rel="stylesheet" href="lounge.css">
    <script type="text/javascript" src="lounge.js"></script>
  </head>
  <body>
    <h1>Welcome to Head First Lounge</h1>
    <p>
      <img src="drinks.gif" alt="Drinks">
    </p>
    <p>
      Join us any evening for refreshing <a href="elixirs.html">elixirs</a>,
      conversation and maybe a game or two of Tap Tap Revolution.
      Wireless access is always provided; BYOWS (Bring Your Own Web Server).
    </p>
  </body>
</html>
```

← 这些都是摘自Head First Lounge的普通HTML 4.01，你可能还记得《Head First HTML》中的这些内容（如果不记得，也不用担心，实际上并不需要你记得）。

BRAIN POWER 看看写HTML5有多容易！

先来看看这个HTML，这是用HTML 4.01（前一个版本）而不是HTML5写的。仔细查看每一行，回忆一下每一部分做什么用。你可以在书上做些标注。下面几页我们会告诉你如何把这个代码转换成HTML5。

pin5i.com 拼吾爱

Sharpen your pencil

仔细查看第2页上的HTML代码之后，你能看出来哪些标记可以用HTML5修改？或者你希望做哪些修改？这里可以帮你指出一点，即doctype定义：

这是“html”的doctype，说明我们没错，正是HTML！

这表示这个标准可以公开得到。

这一部分指出我们在使用HTML 4.01，另外这个标记用English（英文）编写。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
      "http://www.w3.org/TR/html4/strict.dtd">
```

这里指出了定义这个标准的文件。

要记住，**doctype**定义要放在HTML文件的最前面，可以告诉浏览器你的文档是什么类型，在这里，文档类型为HTML 4.01。通过使用doctype，浏览器就能更准确地选择用什么方式来解释和表现你的页面。所以强烈推荐使用doctype。

那么，运用你的推理能力，你认为HTML5的**doctype**定义会是什么样？请写在下面（稍后介绍这个内容时可以再返回来检查你的答案）：

....

....

....

....

....

你的答案写在这里。

pin51.com 最新最全书籍资源下载

HTML5-o-Matic介绍，着手更新你的HTML！

第1步

第1步会让你大吃一惊：跟我来。下面先从Head First Lounge HTML最前面入手，我们要更新doctype，让它有HTML5的新“面貌”。

以下是原来的HTML 4.01版本的doctype：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
```

现在你可能会猜测，我们是不是打算把doctype中的每一处“4”都替换成“5”，对不对？嗯，猜错了。下面就是让人惊奇的地方了，HTML5的新doctype很简单：

```
<!doctype html>
```

要对那些费很大力气记住4.01 doctype的人说声抱歉了。

不用再搜索Google来记住doctype是什么样，也不用从另一个文件复制粘贴doctype，这个doctype非常简单，你肯定能轻松记住。

不过，请等等，还有呢……

这不仅仅是HTML5的doctype，这也是HTML将来所有版本的doctype。换句话说，doctype不会再变了。不仅如此，它甚至在老版本的浏览器中也能正常工作。

制订W3C HTML标准的人向我们承诺，这一次他们确实来真格的了。

第2步

如果你是“Extreme Makeovers”或“The Biggest Loser”电视剧的粉丝，肯定会喜欢第2步。这一步中我们要处理meta标记的内容……来看调整前后的变化：

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
```

调整前(HTML 4)

```
<meta charset="utf-8">
```

调整后(HTML5)

确实，新的meta标记减重不少，也简单得多。用HTML5指定meta标记时，只需随标记提供一个字符编码。不管你是否相信，所有浏览器（包括原来的和新的浏览器）都认识这个meta描述，所以任何页面上都可以使用，它都能正常工作。

pin51.com 拼吾爱

第3步

现在来看第3步，完成这一步后就万事俱备了。这里我们还是要关注<head>元素，打算升级link标记。现在的link标记如下，这是一个类型为text/css的链接，指向一个样式表：

```
<link type="text/css" rel="stylesheet" href="lounge.css">
```

← 老版本！

要升级到HTML5，需要把type属性去掉。为什么呢？因为已经宣布CSS作为HTML5的标准样式，这也是HTML5的默认样式。所以删除type属性之后，新的链接形式如下：

```
<link rel="stylesheet" href="lounge.css">
```

← HTML5

额外奖励

因为你接受很快，所以要特别给你一份额外奖励。通过简化script标记，可以让你的日子更好过。对于HTML5，JavaScript现在已经成为标准，同时也是默认的脚本语言，所以同样可以从script标记中去除type属性。去除type属性后的新script标记如下所示：

```
<script src="lounge.js"></script>
```

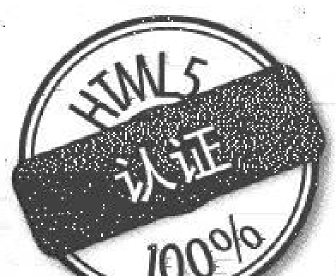
← 即使你对script标记还不太了解，也不用担心，稍后就会介绍……

或者，如果你有一些内联代码，还可以像这样写script标记：

```
<script>
    var youRock = true;
</script>
```

← 所有JavaScript代码都放在这里。

← 后面还会进一步介绍JavaScript。



祝贺你，你已经通过认证，
有能力将任何HTML升级到HTML5！

作为一个经过培训的HTML5-o-Matic用户，你已经掌握了所需的全部技能，可以把任何合法的HTML页面更新为HTML5。现在该动手实践了！

pin51.com 拼吾爱 最新最全资源分享下载站



好啦，好啦，你知道的。到目前为止，我们一直在讨论怎么更新你原来的HTML页面，使它们也能利用HTML5提供的所有特性。可以看到，如果你熟悉HTML 4.01，则肯定会轻车熟路，因为HTML5就是HTML 4.01的一个超集（从实用角度讲，就是HTML 4.01中的所有特性在HTML5中仍得到支持），你要做的就是了解如何指定doctype以及<head>元素中的其余标记，着手使用HTML5。

不过，你讲的也对，我们说得过于简单了，HTML5当然不只是更新这几个元素，它还包括很多其他内容。实际上，真正让所有人兴奋的是能够构建交互性的富页面（或者甚至是复杂的Web应用），为做到一点，HTML5提供了一整套技术，可以配合HTML5标记语言使用。

不过，先别着急。具体介绍这些内容之前，还要做点工作，确保我们的标记没问题。

pin51.com 拼吾爱 最新免费下载

Sharpen your pencil

没想到吧，HTML5标记居然离你这么近!

下面是一些需要更新的老式HTML。请按照HTML5-o-Matic过程把这个HTML更新为HTML5。可以在书上写写画画，把现有的标记代码划掉，增加你认为需要的新标记代码。我们已经提供了一点帮助，突出显示了需要修改的部分。

完成之后，输入修改后的代码（或者如果你愿意，也可以找到练习文件，直接做出修改），在浏览器中加载这个代码，你就能安心坐下来，欣赏你的第一个HTML5作品了。噢，对了，你会在下一页找到答案。

要下载这本书的所有代码和示例文件，请访问http://wickedlysmart.com/hfhtml5。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Head First Lounge</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <link type="text/css" rel="stylesheet" href="lounge.css">
    <script type="text/javascript" src="lounge.js"></script>
  </head>
  <body>
    <h1>Welcome to Head First Lounge</h1>
    <p>
      <img src="drinks.gif" alt="Drinks">
    </p>
    <p>
      Join us any evening for refreshing <a href="elixirs.html">elixirs</a>,
      conversation and maybe a game or two of Tap Tap Revolution.
      Wireless access is always provided;  BYOWS (Bring Your Own Web Server).
    </p>
  </body>
</html>
```

pin51.com 拼吾爱

Sharpen your pencil Solution

没想到吧，HTML5标记居然离你这么近！

下面是一些需要更新的老式HTML。请按照HTML5-o-Matic过程把这个HTML更新为HTML5。可以在书上写写画画，把现有的标记代码划掉，增加你认为需要的新标记代码。我们已经提供了一点帮助，突出显示了需要修改的部分。

下面给出我们的答案。

以下是更新后的代码：

```
<!doctype html>
<html>
  <head>
    <title>Head First Lounge</title>
    <meta charset="utf-8">
     <link rel="stylesheet"href="lounge.css">
     <script src="lounge.js"></script>
  </head>
  <body>
    <h1>Welcome to Head First Lounge</h1>
    <p>
      <img src="drinks.gif" alt="Drinks">
    </p>
    <p>
      Join us any evening for refreshing <a href="elixirs.html">elixirs</a>,
      conversation and maybe a game or two of Tap Tap Revolution.
      Wireless access is always provided;  BYOWS (Bring Your Own Web Server).
    </p>
  </body>
</html>
```

<!doctype html> ← doctype……

<meta charset="utf-8"> ← ……meta标记……

<link …> ← ……link标记……

<script …> ← ……和script标记。

正是这4行改动，使我们的Head First Lounge Web页面成为真正意义上的HTML5。

不相信？可以试试http://validator.w3.org/ 你会看到，它会验证为HTML5。绝无虚言！

pin5i.com 拼吾爱 最新最全电子书免费下载

there are no Dumb Questions

问： **这在老式的浏览器上也能用？就像这些新的doctype、meta之类的标记？老式浏览器怎么处理这些新语法呢？**

答： 对，靠点聪明，也靠点运气。就拿link和script标记上的type属性为例，如今在HTML5中去掉这个属性是合理的，因为CSS和JavaScript现在已经成为标准（当然也是默认的样式和脚本技术）。不过，事实上，浏览器早已经假定默认使用CSS和JavaScript。所以标准一致，真是碰巧，浏览器多年前就已经对这个新标记标准提供了支持。doctype和meta标记也是一样。

问： **那么新的doctype呢？看起来现在是不是太简单了？甚至连版本或DTD都没有。**

答： 嗯，看起来确实有些神奇，使用了这么多年复杂的doctype之后，现在我们居然可以简单到只指出"我们在用HTML"。事情是这样的：HTML原先基于一个名为SGML的标准，这个标准要求提供复杂格式的doctype和DTD。新标准对SGML有所调整，目的是简化HTML语言，使它更加灵活。所以我们不再需要复杂的格式。不仅如此，正如前面所说，这里还有点运气成分，几乎所有浏览器都只是在doctype中查找HTML，来确保在解析一个HTML文档。

问： **你说它不会再变了，不是开玩笑吧？我觉得对于浏览器来说版本非常重要。为什么不使用<!doctype html5>呢？将来还是有可能有HTML6的，对吧？**

答： doctype的使用意味着浏览器制造商要使用doctype告诉浏览器采用它们自己的"标准模式"表现内容。既然我们已经有了真正的标准，HTML5的doctype完全可以告诉所有浏览器这个文档是标准HTML，不论它是版本5还是版本6，或者其他任何版本。

问： **那么，假设不同的浏览器在某个时期有不同的功能，该怎么处理呢？**

答： 没错，特别是在HTML5得到100%支持之前。我们会在这一章以及整本书中介绍这些方面。

问： **这很重要吗？我完全可以输入一个不带doctype和meta标记的页面，它也能很好地工作。我何必操心它是不是完全正确呢？**

答： 是的，浏览器很擅长忽略HTML文件中的小错误。不过，通过包含正确的doctype和meta标记，就能确保浏览器确实知道你想要什么，而不用乱猜。另外，对于那些使用老式浏览器的人来说，新doctype意味着他们将使用标准模式，这正是你要的。要记住，标准模式是指：浏览器会认为你编写的HTML符合一个标准，所以它会使用相应的规则来解释你的页面。如果你没有指定一个doctype，有些浏览器可能会进入"古怪模式"，以为你的Web页面是为老式浏览器编写的，如果标准不符合，就可能不正确地解释你的页面（或者认为它编写得不正确）。

问： **XHTML怎么了？好像好多年前就说它是未来发展方向。**

答： 原先确实是。不过，后来灵活性超越了严格语法，在这个过程中，XHTML（准确地讲是XHTML 2）夭折了，而从人们编写Web页面的方式（以及浏览器表现页面的方式）来讲，HTML5更能让人接受。不过也不用担心，因为了解XHTML能让你更好地编写HTML5内容（你会在整体上更好地把握HTML5）。顺便说一句，如果你真的喜欢XML，还有一种办法可以用严格的格式写HTML5。以后还会详细介绍……

问： **什么是UTF-8？**

答： UTF-8是一种字符编码，支持很多字母表，包括非西方语言。你过去可能已经见过其他的字符集，不过UTF-8被推选为新的标准。而且，它比以前的字符编码更简短，也更容易记。

pin51.com 拼吾爱 最新最全资源分享下载站

Relax

现在并不要求你了解HTML5。

如果你以前从来没有接触过HTML5，也没有关系，不过你应该用过HTML，还有一些基本内容应当知道，比如说元素、标记、属性、嵌套、语义标记和增加样式之间的区别等。

如果你对这都不熟悉，我们有一个小小的建议（也算个广告吧）：还有一本书专门介绍这个内容《Head First HTML with CSS & XHTML》，你应该读一读。如果你对标记语言还算熟悉，可以粗略翻翻，或者在看这本书时作为一个参考。

我们还在附录中对HTML5标记和CSS3提供了一个简短的指南。如果你只是想快速地概要了解补充的新特性，可以先看看本书最后那一部分。



pin5i.com 拼吾爱 最新最全资源分享下载站



HTML5闪亮登场

本周访谈：
最新版HTML的告白

Head First: 欢迎你，HTML5。整个Web都在谈论你呢。对我们来说，你看上去和HTML 4好像没太大不同。为什么大家还这么兴奋呢？

HTML5: 大家之所以兴奋，那是因为我支持新一代的Web应用和体验。

Head First: 没错，不过再问一句，为什么HTML 4或者传说中的“XHTML”做不到呢？

HTML5: XHTML 2已经穷途末路了。所有写实际网页的人都讨厌它。XHTML就是把我们写页面标记的路重走一遍，让所有已有的页面都过时。我说过，“嘿，看我的，我不仅能干新工作，还能包容已有的全部”。我的意思是，如果一个东西用得挺好，为什么还要重头再来一遍呢？这就是我的哲学。

Head First: 听上去在理。不过，你知道的，有些搞标准的人还在说：Web最好遵循他们的“纯”标准。

HTML5: 要知道，我可不关心那些。我更愿意听写实际网页的人怎么讲——他们怎么用我，我能提供什么帮助？其次是创建Web浏览器的开发人员，最不关心的才是那些搞标准的人。我确实会听他们说什么，不过是在与真正的用户没有分歧的前提下。

Head First: 为什么不关心他们呢？

HTML5: 因为如果用户和创建浏览器的人与这些搞标准的人意见不一，就会争论不休了。幸运的是，制定HTML5规范的人都完全支持我，这也是我们的态度。

Head First: 再回到HTML的上一个版本，你说你是HTML 4.01的一个超集。这意味着你是向后兼容的，对不对？是不是表示你打算继续处理以前版本中的所有不好的设计？

HTML5: 我答应过，会尽最大努力处理以往版本留给我的所有问题。也就是说，这并不表示这是对待我的正确方式。我真希望写Web页面的人能深入学习最新的标准，用最好方式的利用我。这样一来，就能让我充分发挥一技之长。不过，重申一次，我不会完全失败，如果页面没有更新，我也会尽我所能地显示这些老页面。

Head First: 下一个问题是 ……

HTML5: 好了，够了!你一直在问以前怎么怎么样。我们还没有谈到真正重要的东西。就我的标记而言，我个人的任务是包容现有的Web，增加一些新的结构元素，让做Web开发的人日子更好过，另外帮助所有浏览器实现人员支持一致的标记语义。不过，我来这里的真正目的是介绍我的新用途：Web应用……

Head First: ……真抱歉，HTML5，我们的访谈时间到了。非常感谢，下一次访谈一定会讨论你想谈的内容。

HTML5: 嗯，真扫兴!

pin51.com 拼吾爱 最新最全资源分享下载站

真正的HTML5请起立……

好了，谢谢你一直那么耐心，坐在这里听我们讲“HTML5-o-Matic”，相信你肯定在猜HTML5远不止这些吧。如今大街上随处可见这样的标语：有了HTML5就不再需要插件，它能用来做各种事情，从简单的页面到一流的游戏都能胜任，就像甜点外面一层的奶油。对不同人的来说，HTML5似乎都不相同……

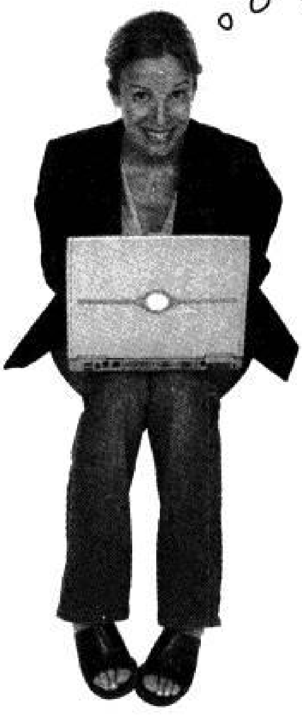














好消息：HTML5囊括了以上全部。人们谈到HTML5时，实际上指的是一个技术“家族”，结合这些技术，就能提供一个全新的模板来构建Web页面和应用。

拼吾爱

pin51.com 拼吾爱

HTML5到底如何工作……

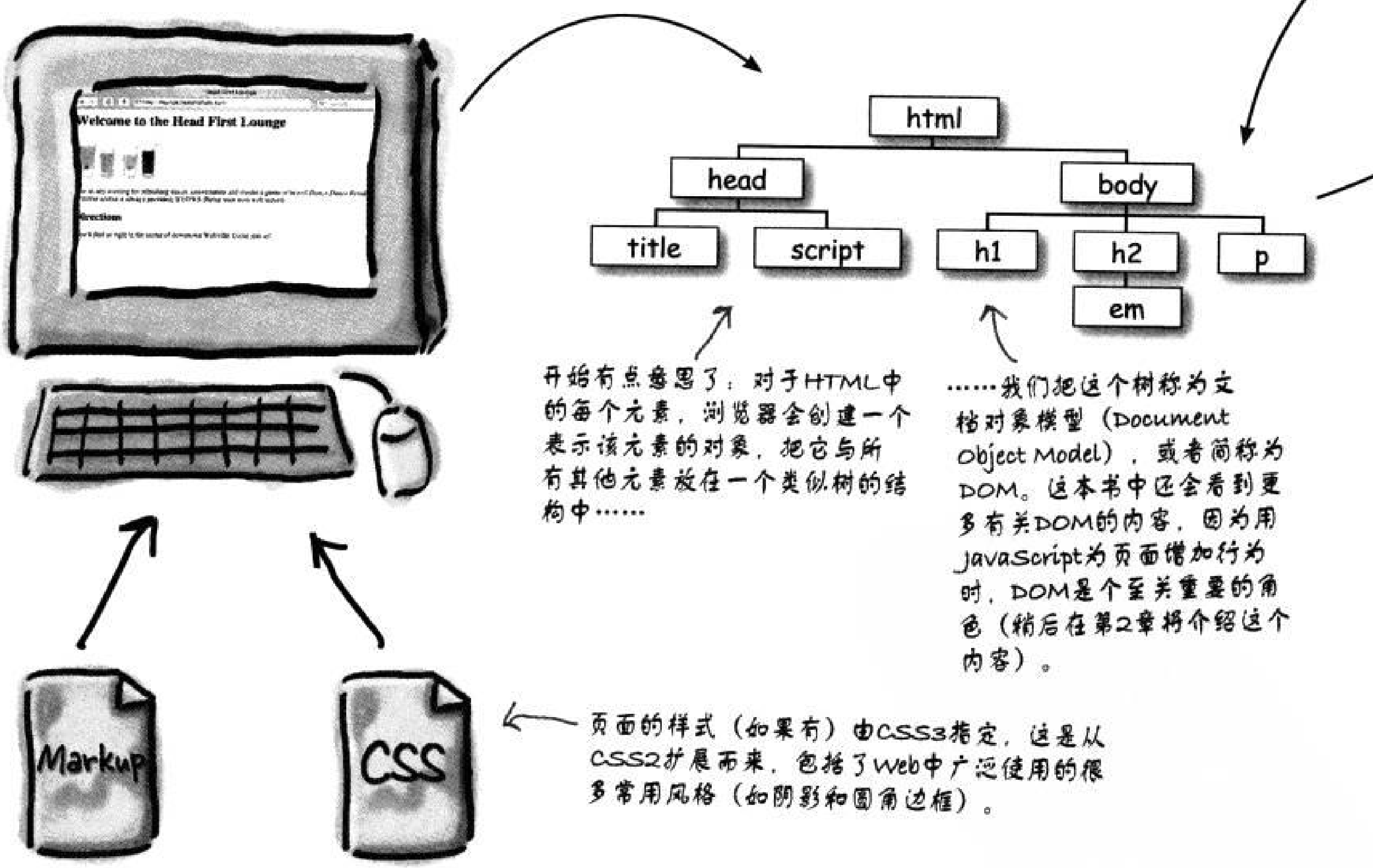
我们说过，HTML5是由一个技术“家族”构成的，不过这到底是什么意思呢？你应该已经知道，HTML标记本身已经扩展为包括有一些新的元素；另外通过CSS3对CSS也有大量补充，可以大大增强指定页面样式的能力。此外还有一个高能“充电器”：JavaScript，有一组全新的JavaScript API可供你使用。

附录中你会看到一个不错的Web镇指南，会向你介绍这些新的HTML5标记和CSS3属性。

下面来看看幕后的工作，了解这些技术如何组合在一起：

1. **浏览器**加载一个文档，其中包括用HTML写的标记和用CSS写的样式。

2. 浏览器加载页面时，还会为**你的文档创建**一个内部模型，其中包含HTML标记的所有元素。



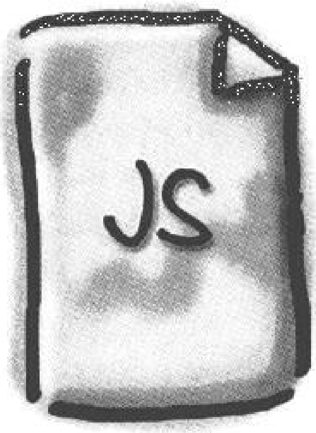
开始有点意思了：对于HTML中的每个元素，浏览器会创建一个表示该元素的对象，把它与所有其他元素放在一个类似树的结构中……

……我们把这个树称为文档对象模型（Document Object Model），或者简称为DOM。这本书中还会看到更多有关DOM的内容，因为用JavaScript为页面增加行为时，DOM是个至关重要的角色（稍后在第2章将介绍这个内容）。

页面的样式（如果有）由CSS3指定，这是从CSS2扩展而来，包括了Web中广泛使用的很多常用风格（如阴影和圆角边框）。

在HTML5中，标记有一些改进，比如<head>元素中的标记，另外还增加了一些元素可供使用（本书会介绍补充的一些元素）。

在幕后



3 浏览器加载页面时，还会加载JavaScript代码，通常页面加载之后就开始执行这些代码。

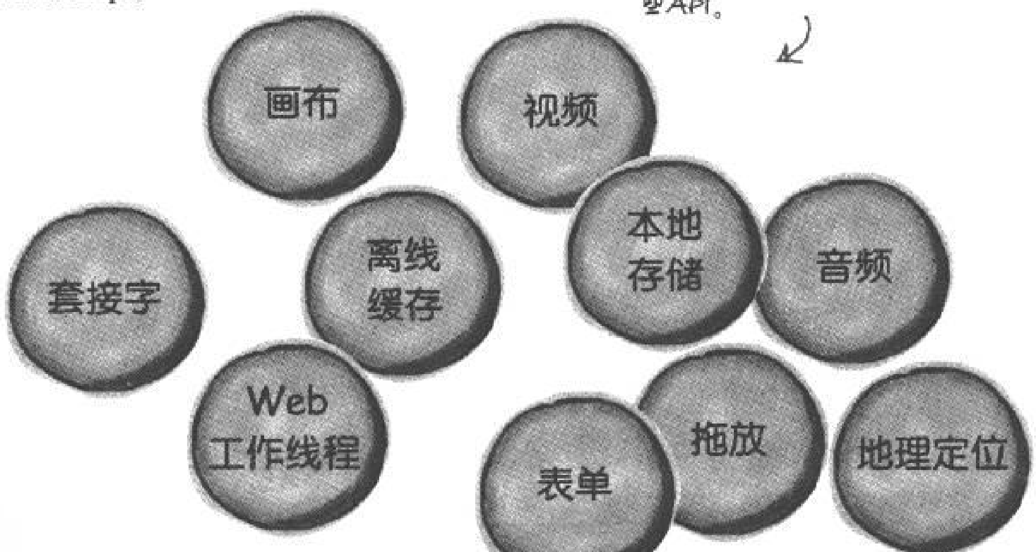
利用JavaScript，可以通过处理DOM与页面交互、对用户或浏览器生成的事件做出响应，或者使用所有新的API。

JavaScript通过DOM与页面交互。

4 通过这些API，你可以访问音频和视频、使用画布完成2D绘图、访问本地存储，还可以使用构建应用所需的很多其他优秀技术。要记住，要想使用所有这些API，需要用到JavaScript。

API也称为应用编程接口（Application Programming Interfaces），提供了一组对象、方法和属性，可以用来访问这些技术的所有功能。这本书中将介绍其中一些API。

认识JavaScript API



pin51.com 拼吾爱 最新最全 大家站

WHO DOES WHAT?

我们一直在说“技术家族”，让人感觉它们好像就是一个家族。不过我们还没有正式认识这个家族的成员，现在就来认识一下吧。你会在下面看到这个家族的大部分成员，看看你能不能搞清楚究竟谁做什么。我们已经帮你确定了一个。不用担心，我们知道这是你第一次接触HTML5家族成员，所以本章最后会给出答案。

CSS3	用我你就能直接在Web页面上绘图。有了我，你可以绘制文本、图像、线条、圆、矩形、图案和梯度。我会让你成为真正的艺术家。
Web工作线程	你在HTML 4中可能用我输入过信息，不过在HTML5中我变得更出色。我可以要求你填写域，可以更容易地验证你是不是在指定的位置键入了Email、URL或电话号码。
表单	以前你需要插件才能支持我们，不过现在我们已经是HTML元素大家庭的直系成员。想看点什么或者听点什么吗？你会需要我们的。
离线Web应用	我们存在的意义就是帮助明确页面的结构和语义含义，包括提供一些新方法在页面中建立分区、页眉、页脚和导航。
音频&视频	我是这个家族里最时髦的。你以前可能用过我，不过你知道吗？现在我可以让你的元素运行动画，给它们最漂亮的圆角边框，甚至可以加阴影！
新元素	可以在每个用户的浏览器里把我当做本地存储来使用。是不是需要存储一些首选项或一些购物车商品？或者是不是为了提高效率想要存放一个庞大的缓存？我就是你想要的API。
本地存储	是不是希望在未连接网络时应用也能正常工作？我能助你一臂之力。
画布	我能告诉你在哪里，还可以与Google maps很好地合作。
地理定位	如果你需要多个脚本在后台并发运行，让你的用户界面保持响应性，你就会想到我。

（已连线：表单 → “你在HTML 4中可能用我输入过信息……”）

pin51.com 拼吾爱

CASE FILE: HTML5

你的任务……

……（如果你接受这个任务）是对所有HTML浏览器做一些侦察。你肯定听说有些浏览器已经支持HTML5，有些还不支持。我们需要你仔细了解，因为事实上……

TOP SECRET

第一项任务：浏览器侦察

请四处走访一下，确定[redacted]各个浏览器的支持水平（提示，可以到这里查找有关的一些资源：HTTP://WWW.WICKEDLYSMART.COM/HFHTML5/BROWSERSUPPORT.HTML，[redacted]这里假设都使用浏览器的最新版本。对于每组浏览器/特性，如果得到支持就画勾，然后自己为各个浏览器对HTML5的支持水平打分[redacted]等你完成任务，要给出报告作为你的下一个任务！

浏览器 \ 特性	视频	音频	画布	Web存储	地理定位	Web工作线程	离线Web应用
Firefox							
Safari							
Chrome							
Mobile WebKit							
Opera							
IE 6, 7							
IE 8							
IE 9							

iOS和Android设备（以及其他移动设备）→ Mobile WebKit

www.pin51.com 下载站 拼吾爱 最新

第一项任务：浏览器侦察答案

TOP SECRET

CASE FILE: HTML5

我们的答案并不真实，这是假想为2015年填写的。你的答案应当反映你读这本书当时的情况。不过我们希望你能看到未来的趋势。

浏览器 \ 特性	视频	音频	画布	Web存储	地理定位	Web工作线程	离线Web应用
Firefox							
Safari	✓	✓	✓	✓	✓	✓	✓
Chrome	✓	✓	✓	✓	✓	✓	✓
Mobile WebKit	✓	✓	✓	✓	✓	✓	✓
Opera	✓	✓	✓	✓	✓	✓	✓
IE 6, 7							
IE 8				✓			
IE 9	✓	✓	✓	✓	✓		

尽管签署、密封和交付标准还需要一段时日，不过早在这之前你就能用到充分支持HTML5的浏览器。实际上，很多特性在现代浏览器上已经得到了广泛支持。这也是为什么现在就该开始使用HTML5的原因。另外还有一个好处，如果你现在就着手使用HTML5，你掌握的前沿技术肯定会让你身边的朋友和同事羡慕不已。

而且很快工资也会涨！

pin51.com 拼吾爱 最新最全资源分享下载站



别急，深呼吸。

首先，HTML5是HTML的一个超集，所以你的目标肯定是只写一个HTML页面。你说得没错，任何浏览器支持的特性都可能不同，这取决于当前的浏览器是什么，还要看用户对升级是否积极。所以，一定要记住，HTML5的一些较新的特性可能未能得到支持，这就回到了你的问题上来：如何处理这种情况。

目前，HTML5的底层设计原则之一就是允许你的页面妥善地降级。这说明，如果用户的浏览器未能提供一个新特性，你就应当提供有一个有意义的候选功能。在这本书中，我们会告诉你如何编写页面来做到这一点。

不过有一个好消息，所有浏览器都在朝着HTML5标准和相关技术的方向前进（甚至包括移动浏览器），所以过一段时间，妥善降级可能不再是一个规则，而更应算是一种例外情况（尽管你总希望尽量为用户提供一种有意义的体验，而不论他们使用哪一个浏览器）。

nin51.com 拼吾爱

there are no Dumb Questions

问： 我听说HTML5标准在2022年前都不会成为最终推荐的标准！是真的吗？如果是这样，还有什么必要了解它呢？

答： W3C是正式推荐HTML5标准的标准组织，对于W3C，你要知道他们是一个保守的团体，有多保守呢？可以这么讲，他们甚至宁愿等到几代HTML5浏览器出现又消失后才会签署这个标准。没关系，过几年标准就会签订，浏览器制造商也正在努力实现这个标准。所以，尽管你说的没错，HTML5成为"最终推荐标准"可能还有段时间，不过2014年前应该会有一个稳定的标准，而且你需要的所有实用功能现在HTML5都能提供。

问： HTML5成为最终标准后会怎么样呢？

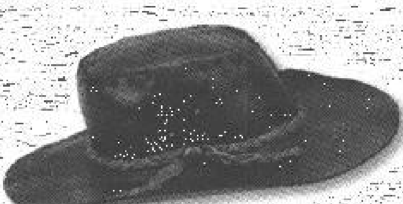
答： HTML6？我们不清楚，不过不管会是什么，基本的都不变。要记住，即使我们确实采用了HTML6，doctype也不会改变。假设W3C信守承诺，将来版本的HTML保持向后兼容。不论下一个版本是什么，我们都能轻车熟路。

问： Chrome、Safari、Firefox，还有一大堆移动浏览器……这个世界是不是变得越来越混乱了？怎么能确保我们的页面在所有这些浏览器上都能用呢？

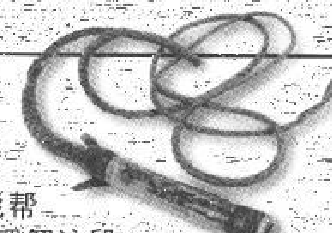
答： 市场上对于浏览器（包括桌面浏览器和移动浏览器）确实存在大量良性竞争，但实际上，其中很多浏览器都基于一些通用的HTML引擎。例如，Chrome、Safari，以及Android和iPhone上的移动浏览器都基于WebKit，这是一个开源的浏览器引擎。所以，总的来讲，你的页面完全可以顺利地在多个浏览器上运转，根本不用你做太多工作。

问： 为什么不干脆使用Flash来解决跨浏览器的问题呢？

答： 对于很多应用来说，Flash是一个很不错的工具，在桌面领域，它确实广泛应用在众多操作系统和浏览器上。HTML5及其技术"家族"努力保证你能用开放标准完成Flash能做的同样的事情。所以，你会选择哪条路呢？有一点可供考虑，要知道，Google、Apple、Microsoft和众多其他厂商都对HTML5技术做了大量投入。从长远来讲，HTML5肯定会成为一个"巨星"，而且在移动领域，这已经成为事实。所以，尽管选择权在你手里，而且我们相信这两种技术还会并存相当长时间，但业界朝着开放标准的方向前进是不容置疑的。



HTML考古



我们做了一些深入挖掘，发现了嵌入在一个HTML页面中的一些代码。希望你能帮助我们破解这些代码，搞清楚它到底是什么意思。不用担心，我们并不指望你理解这段代码，只是想让你做一些推理，让头脑开动起来……

```
<script>
    var walksLike = "duck";
    var soundsLike = document.getElementById("soundslike");
    if (walksLike == "dog") {
        soundsLike.innerHTML = "Woof! Woof!";
    } else if (walksLike == "duck") {
        soundsLike.innerHTML = "Quack, Quack";
    } else {
        soundsLike.innerHTML = "Crickets...";
    }
</script>
```

提示：document表示整个HTML页面，getElementById可能与HTML元素和id有关。

pin51.com 拼吾爱 最新最全资源分享下载站



我们得谈一谈了。

如果你看过我们的《Head First HTML & CSS》(或者更准确地讲，如果你仔细读过这本书，而没有把它当柴火烧掉），相信你可能已经很好地掌握了如何使用标记语言和样式表创建漂亮的Web页面。了解这两个技术会对你很有帮助……

不过，对于HTML5，情况发生了改变：Web页面已经变身为一种富体验（以及成熟的应用），不仅有行为，可以动态更新，还可以与用户交互。构建这种页面需要一些编程，如果你打算为浏览器编写代码，只有一条路：JavaScript。

如果你以前做过编程或者写过简单的脚本，你会很轻松：JavaScript（尽管有一些不好的传言）是一种绝妙的语言，本书中，我们会带你了解编写应用需要知道的全部内容。如果你以前从未编写过程序，我们也会尽全力带你上路。不论哪种情况，都会发现JavaScript的一大好处：初学的程序员可以很容易地上手。

我们实在想不出还有什么更好或者更有趣的方法来学习编程！

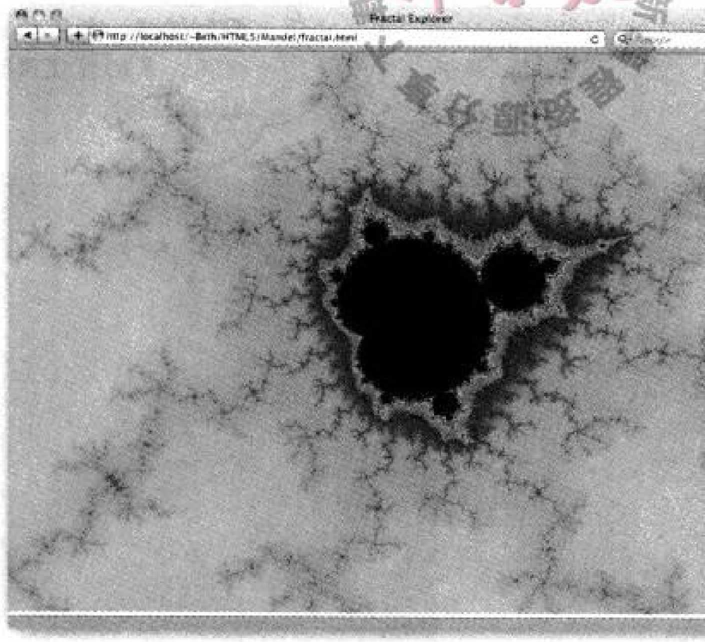
那么现在做什么呢？先来简要介绍一些JavaScript知识，然后在第2章再深入研究。实际上，对现在来讲，先不要纠结于后面几页的每一个细节，我们只是希望你对JavaScript有点感觉。

用JavaScript能做些什么？

JavaScript为你的Web页面打开了一个新的世界，提供了丰富的描述和功能。下面来看利用JavaScript和HTML5可以做的一些事情……

采用新方法与页面交互，这些方法同时适用于桌面和移动设备。

利用HTML5 & JavaScript，你可以直接在页面上创建一个可绘制的2D表面，根本无需插件。



使用Web工作线程可以提高JavaScript代码的效率，完成一些复杂的计算，或者使你的应用更具响应性。甚至可以更好地利用你的用户的多核处理器！

使页面掌握位置信息，知道你的用户所在的位置，向他们显示附近有些什么，带他们进行目标排查，指明方向，或者把有共同兴趣的人汇集到同一区域。



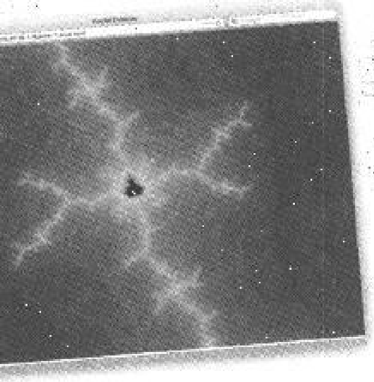
访问任何Web服务，并将数据（几乎实时地）传回应用。

使用浏览器存储在本地缓存数据，提高移动应用的速度。

不再需要特殊的插件来播放视频。

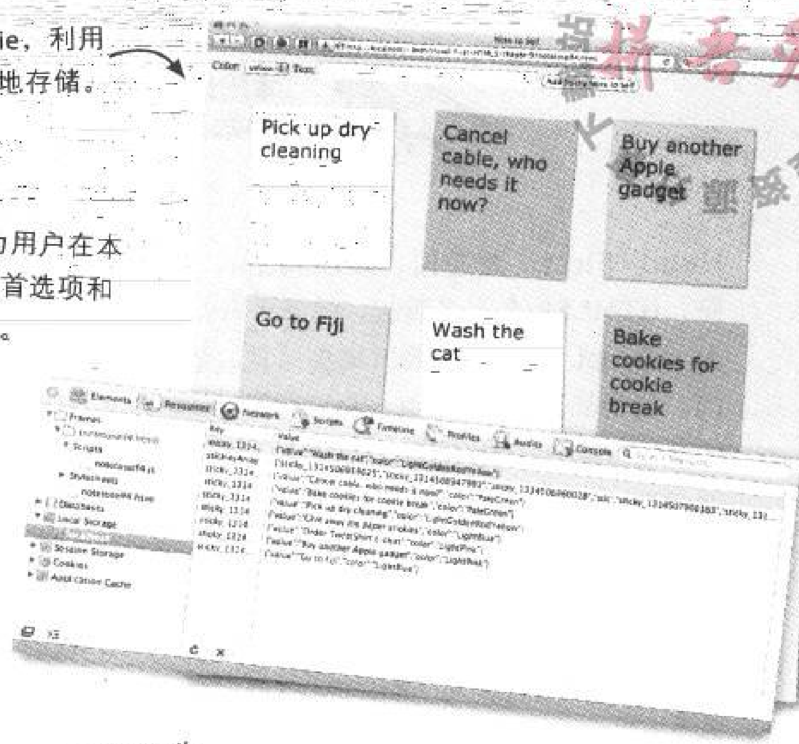
将你的页面与Google Maps集成，甚至允许用户实时地跟踪他们的移动轨迹。

使用HTML和JavaScript创建你自己的视频回放控件。



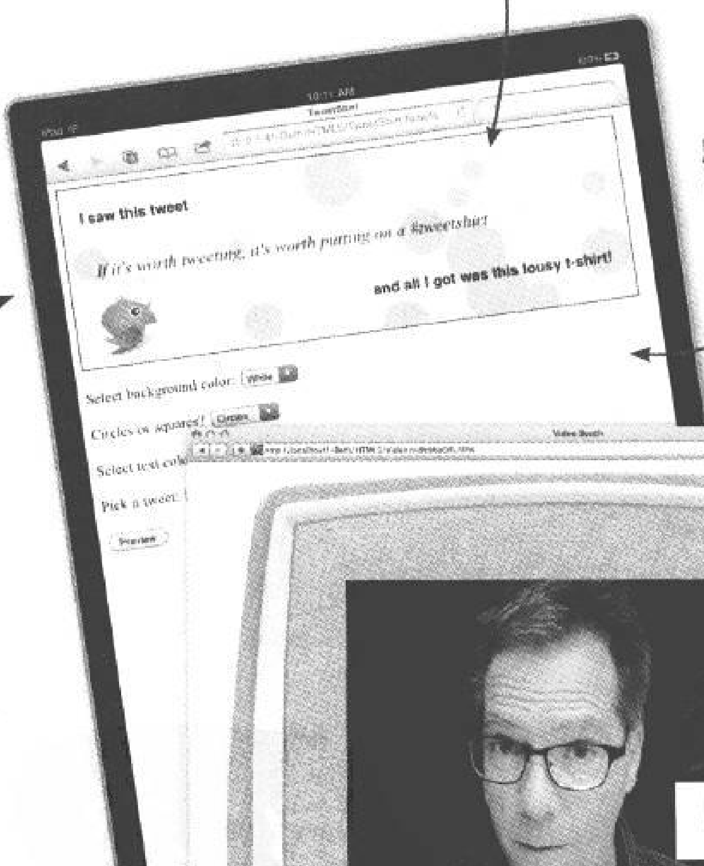
告别浏览器Cookie，利用基于浏览器的本地存储。

利用JavaScript，你可以为用户在本地（浏览器中）存储大量首选项和数据，甚至可以离线访问。



浏览器肯定不再只是应付枯燥的文档了。有了JavaScript，你可以直接在浏览器上画图。

用JavaScript可以让你的表单如虎添翼，提供真正的交互性。



建立完整的视频体验，可以采用新的方式加入视频。



使用JavaScript的强大功能，可以在浏览器中完成完备的视频处理。不仅能创建特效，甚至可以直接处理视频像素。



你可能以为我们肯定是在网上查了很多才找到这样一些最让人兴奋的例子，是不是？你错了。我们只是拿出了本书中几个例子的截屏图。是不是惊到了？居然这么酷！既然我们已经身处Web镇，就要学会这里的方言：JavaScript。来吧，出发吧。

pin5i.com 拼吾爱



JavaScript闪亮登场

本周访谈：
脚本语言的告白

Head First: 欢迎你，JavaScript。很高兴你能从百忙中抽出时间来作客。我们开门见山吧，HTML5如今声名鹊起，你怎么看？

JavaScript: 我可不爱出风头，我就是默默躲在幕后的那种人。这么说吧，很多功劳本来是我的，可是都算到了HTML5头上。

Head First: 怎么这么说？

JavaScript: HTML5的工作需要一整套技术，比如2D画布、本地存储、Web工作线程，诸如此类。不过事实上，是我，对，就是我JavaScript，在真正使用这些技术。HTML5只是为你提供了一个场所，可以把这些体验融合在一起表现出来，不过如果没有我，你根本不会得到任何有趣的体验。不过没关系，HTML5加油吧，我会继续做我自己的事情。

Head First: 你对新入门的HTML5程序员有什么建议？

JavaScript: 很简单。如果你确实想掌握HTML5，一定要花点时间学习JavaScript，还要了解与HTML5一同使用的所有库。

Head First: 要知道，你的名声可不算太好。1998年有评论这样说："JavaScript最多也只是个半成品，一个憋脚的脚本语言。"

JavaScript: 真让人伤心。尽管我问世的年代还是多事之秋，众多编程语言还没有形成一个清楚明了的规范环境，但是一直以来我都是使用最广泛的编程语言之一，所以我不会那么快丧失信心。不仅如此，已经补充了大量资源着力使我更健壮，行动更迅速。与十年前相比，我至少快了100倍。

Head First: 太让人佩服了。

JavaScript:噢，对了，你可能还没听说，制订标准的人已经告诉我现在我是HTML5的默认脚本语言。所以，我会一直在这里。实际上，你甚至不用再在<script>标记中明确指出"JavaScript"。1998年他们可能认为我憋脚，不过你再看看，如今还能不能找到JScript、VBScript、Java Applets和所有那些失败的浏览器语言的踪影？

Head First: 嗯，听上去你确实是创造一流HTML5体验的关键。不过，对你的评价真是毁誉参半。

JavaScript: 尽管有些负面的传言，但我是一种功能非常强大的语言，所以你确实应当花些时间来好好学习我。另一方面，我很流行，因为JavaScript的上手和使用都非常容易。我在这两方面都有绝对优势，不是吗？

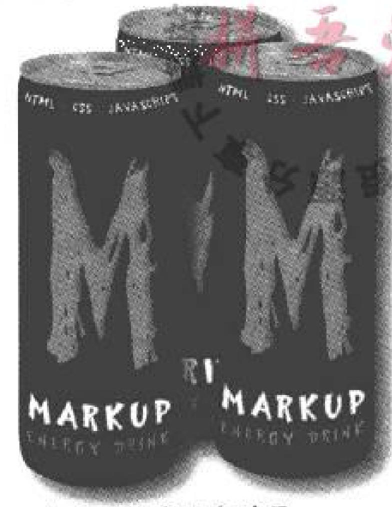
Head First: 听起来不错！谢谢你，JavaScript，感谢你与我们交流。

JavaScript: 我也很荣幸。

pin5i.com

编写正式的JavaScript

谈论了这么久JavaScript，相信你肯定想深入了解它到底是什么。否则这本书也不会叫做Head First。下面会给出一个超级正式的商业应用。对现在来说，你先仔细看看代码，找点感觉。你认为每行代码要做什么？请写下来。不用担心，我们并不指望你现在就无所不知，不过相信你肯定能对这些代码的工作猜得八九不离十。另外，写完之后，翻到下一页，可以看看你猜得对不对……



把你的答案写在这里。

这里换成你最喜欢的饮料。

```
var drink = "Energy Drink";
var lyrics = "";
var cans = 99;

while (cans > 0) {
    lyrics = lyrics + cans + " cans of "
            + drink + " on the wall <br>";
    lyrics = lyrics + cans + " cans of "
            + drink + "<br>";
    lyrics = lyrics + "Take one down, pass it around,<br>";

    if (cans > 1) {
        lyrics = lyrics + (cans-1) + " cans of "
            + drink + " on the wall <br>";
    }

    else {
        lyrics = lyrics + "No more cans of "
            + drink + " on the wall <br>";
    }
    cans = cans - 1;
}
document.write(lyrics);
```

pin51.com

编写正式的JavaScript（续）……



再仔细读读这个代码，看看你猜对了没有。现在只是希望你对代码有些感觉，稍后就会逐步地详细介绍。

```
var drink = "Energy Drink";

var lyrics = "";
var cans = 99;


while (cans > 0) {


    lyrics = lyrics + cans + " cans of "

              + drink + " on the wall <br>";

    lyrics = lyrics + cans + " cans of "

              + drink + "<br>";
    lyrics = lyrics + "Take one down, pass it around,<br>";

    if (cans > 1) {

        lyrics = lyrics + (cans-1) + " cans of "
                 + drink + " on the wall <br>";
    }
    else {

        lyrics = lyrics +  "No more cans of "

                 + drink + " on the wall <br>";
    }
    cans = cans - 1;
}


document.write(lyrics);
```

- 声明一个变量，并将它赋值为“Energy Drink”。
- 声明另一个变量，把它赋值为一个空串。
- 再声明一个变量，赋为一个数值99。
- 这是一个while循环。它指出，只要cans的数目大于0，就反复做大括号之间的工作。没有cans时（cans为0）则停止循环。
- 使用字符串连接操作符“+”，把下一行歌词增加到变量lyrics。
- 用一个HTML换行符结束这一行。
- 再做同样的工作——毕竟，歌曲都是这样的，对吧？
- 增加下一句，同样要使用字符串连接。
- 如果还剩余饮料罐（也就是说，cans大于1）……
- ……增加最后一行。
- 否则，不再有剩余的饮料罐……
- ……所以在歌词最后增加“No more cans”。
- 将剩余的cans数目减1。
- 我们已经把这首歌的每一行都保存到变量lyrics中，所以现在告诉Web页面将歌词写出来，这说明这个字符串会增加到页面，使你能看到这首歌。

pin51.com 拼吾爱

运行测试

你是不是觉得这个练习很难，但还是没有真正上手试一试JavaScript，是吧？现在你要做的就是把上一页的代码（连同下面的HTML）输入到一个文件中（比如index.html），然后在浏览器中加载这个文件。你会看到下面的结果：

记住，要下载本书的所有代码和示例文件，请访问http://wickedlysmart.com/hfhtml5。

输入这个HTML。

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My First JavaScript</title>
  </head>
  <body>
    <script>

    </script>
  </body>
</html>
```

包围JavaScript代码的<script>和</script>标记，告诉页面它们之间包围的内容是JavaScript，而不是HTML。

在这里输入上一页的JavaScript代码。

这是运行这个代码的测试结果。这个代码会为墙上的99瓶罐装啤酒功能性饮料创建完整的歌词，并把文本写入浏览器文档。



pin51.com 拼吾爱 最新最全 绿色软件下载站

there are no Dumb Questions

问：为什么这个HTML体中除了脚本没有其他内容？

答： 我们选择从一个空的HTML体开始，因为这个页面的所有内容都是使用JavaScript代码创建的。当然，也可以直接在body元素中输入这些歌词（工作量可能不小），或者可以让代码帮我们完成这个艰巨的任务（我们正是这样做的），然后让代码用document.write将歌词插入到页面中。

要记住，这里只是冰山一角，本书还会用很多篇幅来介绍如何用代码为页面填入内容。

问：我们构建了这首歌的全部歌词，这一点我明白了，不过document.write做什么呢？另外，文本是怎么放到文档中的？

答： 嗯，document.write会取一个文本串，把它插入到文档中。实际上，它会在script标记所在的位置输出这个串。所以，在这里，document.write将把文本串放在页面体中。

接下来你就会看到，利用JavaScript，可以采用更多复杂的方式修改一个动态文档的文本，不过，这个例子只是让你感受代码如何动态地改变一个页面。

问：你反复提到Web页面和Web应用，这两个词有什么不同吗？怎么才算是一个web应用呢？

答： 这个问题问得好，因为我们的提法并不严格。这两个词之间没有绝对的差别；换句话说，要把一个用HTML、JavaScript和/或CSS编写的页面变成一个Web应用，并不需要你做任何特殊的工作。所以只是看法不同而已。

如果一个页面表现得像是一个应用，而不只是一个静态文档，我们就可以认为它是一个Web应用，而不仅仅是Web页面。我们认为应用应该具备一些特点，比如维护大量状态、管理与用户更复杂的交互、无需页面刷新就能显示不断更新的动态数据，或者甚至可以完成更复杂的任务或计算。

问：嘿，JavaScript是很棒，不过CSS呢？我早都听说CSS3的一些新玩艺了，我真是迫不及待地想用上这些新特性，好让我的页面看上去更漂亮。

答： 是啊，CSS有了长足的发展，它与HTML5能合作得这么好也让我们很震惊。尽管这本书不是专门介绍CSS3，不过我们肯定会充分利用它的一些新功能的。你可能知道，原先在HTML中利用图像增加圆角边框和阴影时，以及利用JavaScript增加简单的动画时，需要很多技巧，而现在所有这些利用CSS3轻轻松松就能办到。

所以，没错，这本书确实要利用CSS3的强大功能，用到时我们会特别指出。

pin5i.com 拼吾爱 最新资源分享下载站

我们已经谈到很多内容，包括HTML标记、JavaScript API、“技术家族”，还有CSS3。到底什么是HTML5？能让大家如此兴奋肯定不只是因为标记吧……



下面给出我们的非官方答案：

标记 + JavaScript API + CSS = ~~奇妙美味~~ HTML5

要知道，大多数人谈论HTML5的种种承诺时，实际上就是指所有这些技术的结合。也就是说，我们会用标记来搭建页面的核心结构，用JavaScript和所有相关API增加行为和新功能，另外用CSS为页面增加样式，它们结合在一起，这些就是我们用来构建未来Web应用的技术。

那么，为什么说这是非官方的说法呢？嗯，有些人喜欢严格区分这些技术，并区别它们分别属于哪个标准。这很好，也是有意义的。不过，我们真正关心的是：浏览器提供了哪些技术，能不能用来制作我们的页面和应用？所以，我们说HTML5就是标记 + JavaScript API + CSS，而且我们认为，人们作为一个技术谈论HTML5时，通常就是这个意思。

如果你确实想知道这些技术如何集成在一起构成一组标准（我们都会对此感兴趣），建议访问w3.org来了解更多信息。

pin51.com 拼吾爱



祝贺你，你已经读完了第1章，而且编写了你的第一个HTML5！

还有你的第一个JavaScript代码！

翻到下一章之前，你还要带些任务回去。用下面的磁贴填写以下公式，回答问题：“什么是HTML5？”要当心，这堆磁贴里特意放置了一些扰乱你的诱饵。完成这个问题后，可以休息一下，放松放松，准备进入第2章。

.... + ... + ... = ...

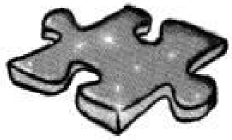
- HTML5
- JavaScript API
- JavaScript
- XML
- CSS
- 画布
- 红酒
- 标记
- Mr. Pibb
- XHTML
- 表单
- CSS3
- 地理定位
- <script>
- 视频
- 文档
- 音频
- Web工作线程
- 离线访问
- 本地存储
- 奇妙美味

www.ain51.com 最新最全的资源分享站 我爱分享

BULLET POINTS

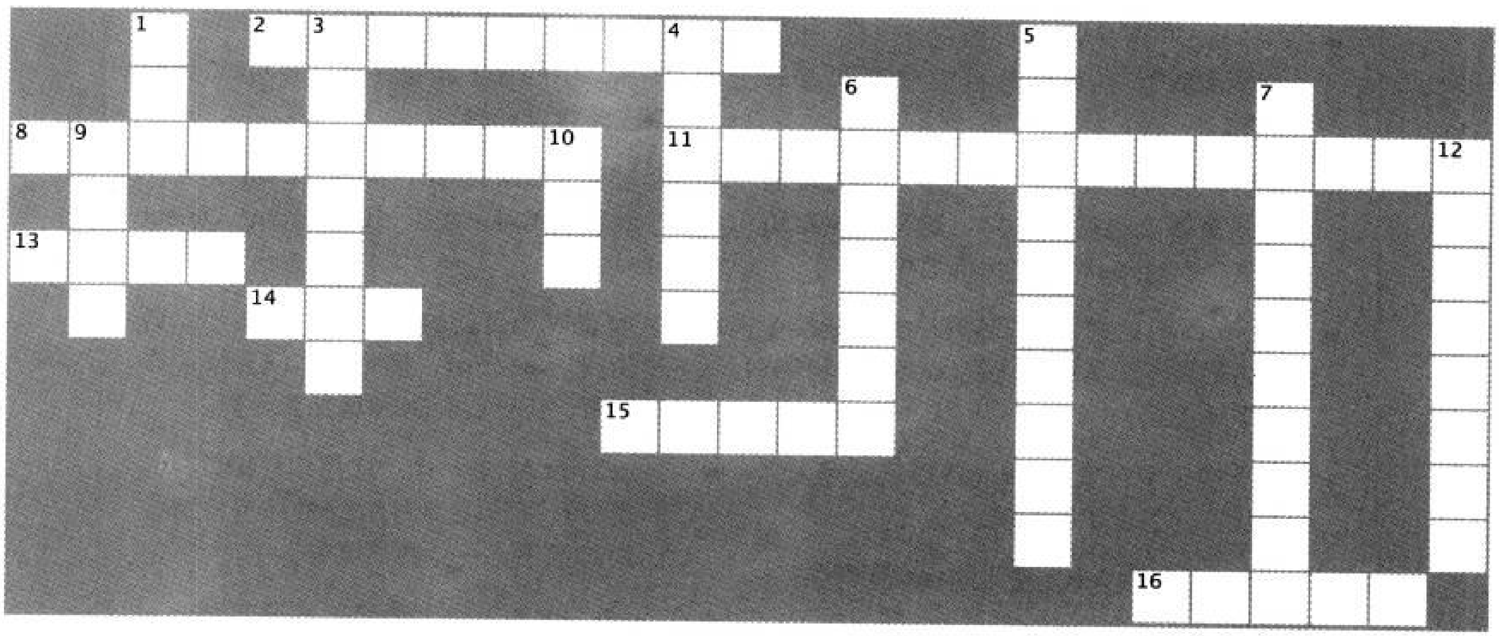
- HTML5是最新版本的HTML。它引入了简化的标记、新的语义和媒体元素，另外要依赖于一组支持Web应用的JavaScript库。
- XHTML不再是Web页面的标准。开发人员和W3C决定还是继续扩展和改进HTML。
- 新的、更为简单的HTML5 doctype在较老的浏览器上也得到支持，这些浏览器看到这个doctype时会使用标准模式。
- <script>标记或指向CSS的样式表链接中不再需要type属性。现在JavaScript和CSS是默认类型。
- 用于指定字符集的<meta>标记已经大为简化，只包含字符编码。
- UTF-8现在是Web上使用的标准字符集。
- 对doctype和<meta>标记做出修改不会影响页面在较老浏览器上的显示。
- HTML5的新元素是HTML 4元素的一个超集，这说明，较老的页面在现代浏览器中仍能正常工作。
- 按官方说法，HTML5标准在2014年前不会正式完成，不过大多数现代浏览器在此之前就能提供支持（现在就有很多浏览器支持HTML5）！
- HTML5引入了一些元素，可以向页面增加新的语义，与HTML 4.01相比，可以提供更多选项来创建Web页面结构。本书并不打算一一介绍，不过会在附录中给出一个简短的指南。
- HTML5中的很多新特性都需要JavaScript来充分加以利用。
- 通过使用JavaScript，可以与DOM交互，也就是文档对象模型（Document Object Model）。
- DOM是Web页面的浏览器内部表示。通过使用JavaScript，你可以访问元素、修改元素，还可以向DOM增加新元素。
- JavaScript API是一个“应用编程接口”。利用API，可以控制HTML5的所有方面，比如2D绘图、视频回放等等。
- JavaScript是世界上最流行的语言之一。最近几年，JavaScript实现有了显著的改进。
- 可以检测一个浏览器中是否支持某个新特性，如果不支持还能够妥善地降级。
- CSS是HTML5的样式标准，很多人用“HTML5”描述创建Web应用所用的技术家族时，都包含CSS。

pin51.com 拼吾爱



HTML5填字游戏

现在让你的右脑休息一下，活动活动左脑。这些词都与HTML有关，而且都能在本章中找到。



横向

2. __________ 的插件也称为垃圾。
8. 通过3个步骤整理HTML5的产品。
11. 你的任务是浏览器__________。
13. HTML5真正的功能是JavaScript ____。
14. JavaScript比十年前要快______ 倍。
15. 使用一个______ 循环打印一首歌的歌词。
16. 收到抬头是“Dear John”的信。

纵向

1. ________是一个Web页面的内部表示。
3. HTML5之前的HTML版本。
4. <________> 标记告诉浏览器后面是JavaScript，而不是HTML。
5. 我们希望Web体验能 ______地降级。
6. 比HTML4.01版本简单得多。
7. HTML5的标准脚本语言。
9. HTML5中link和script标记不再需要这个属性。
10. HTML5的官方样式标准。
12. HTML中的新______ 可以增加语义和结构。

pin5i.com 拼吾爱

WHO DOES WHAT? 答案

我们一直在说"技术家族"，让人感觉它们好像就是一个家族。不过我们还没有正式认识这个家族的成员，现在就来认识一下吧。你会在下面看到这个家族的大部分成员，看看你能不能搞清楚究竟谁做什么。我们已经帮你确定了一个。不用担心，我们知道这是你第一次接触HTML5家族成员，下面给出答案。

- CSS3
- Web工作线程
- 表单
- 离线Web应用
- 音频&视频
- 新元素
- 本地存储
- 画布
- 地理定位

用我你就能直接在Web页面上绘图。有了我，你可以绘制文本、图像、线条、圆、矩形、图案和梯度。我会让你成为真正的艺术家。

你在HTML4中可能用我输入过信息，不过在HTML5中我变得更出色。我可以要求你填写域，可以更容易地验证你是不是在指定的位置输入了Email、URL或电话号码。

以前你需要插件才能支持我们，不过现在我们已经是HTML元素大家庭的直系成员。想看点什么或者听点什么吗？你会需要我们的。

我们存在的意义就是帮助明确页面的结构和语义含义，包括提供一些新方法在页面中建立分区、页眉、页脚和导航。

我是这个家族里最时髦的。你以前可能用过我，不过你知道吗？现在我可以让你的元素运行动画，给它们最漂亮的圆角边框，甚至可以加阴影！

可以在每个用户的浏览器里把我当做本地存储来使用。是不是需要存储一些首选项或一些购物车商品？或者是不是为了提高效率想要存放一个庞大的缓存？我就是你想要的API。

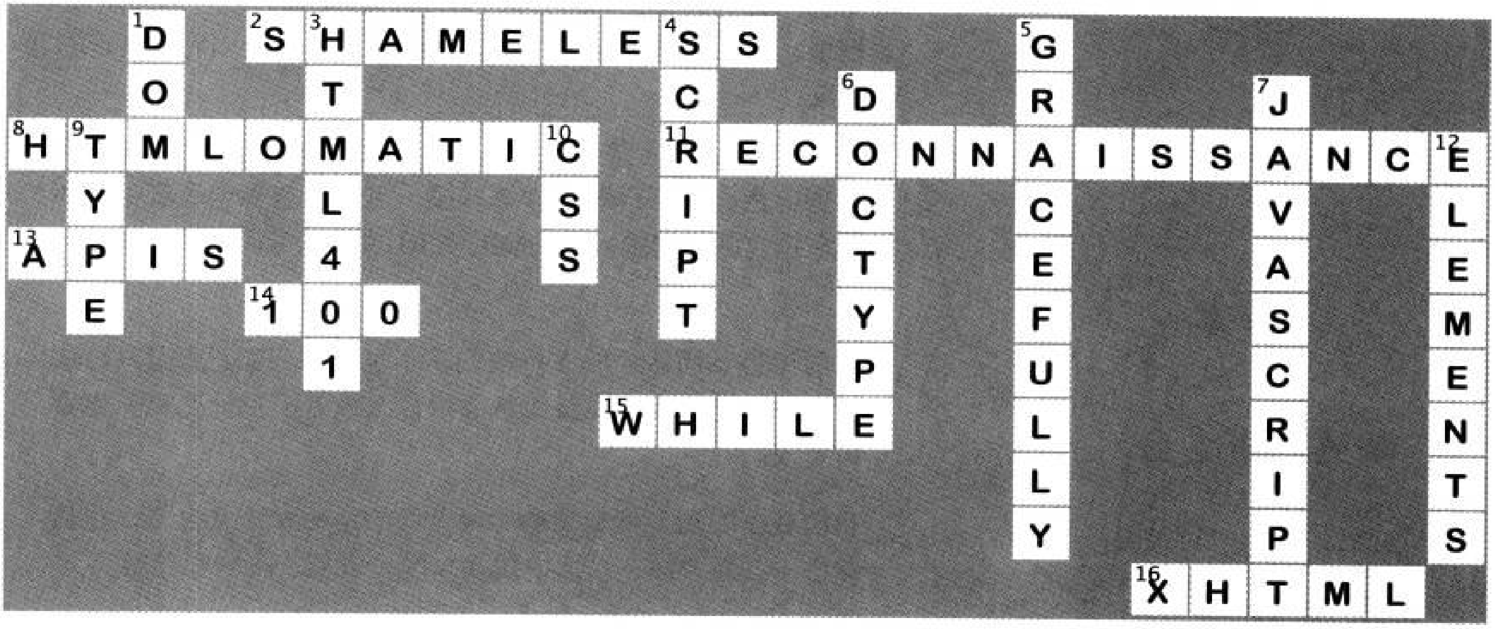
是不是希望在未连接网络时应用也能正常工作？我能助你一臂之力。

我能告诉你在哪里，还可以与Google maps很好地合作。

如果你需要多个脚本在后台并发运行，让你的用户界面保持响应性，你就会想到我。

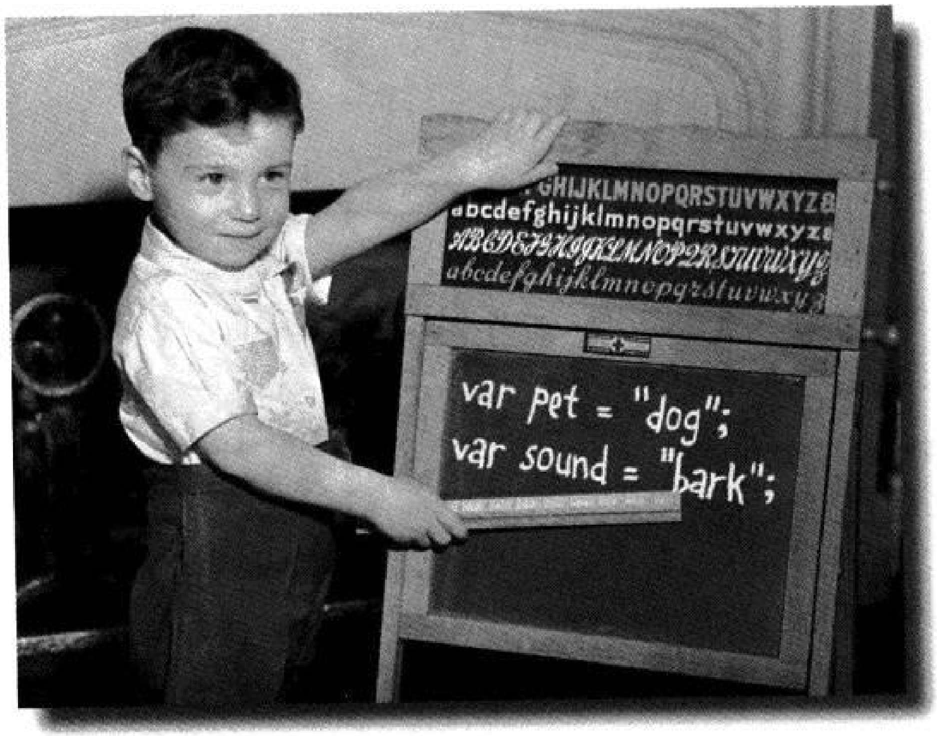
pin5i.com 拼吾爱 最新编程资源分享下载站

HTML5填字游戏答案



2 介绍JavaScript和DOM

一点点代码



JavaScript会带你进入新境界。你已经了解了HTML标记（也称为结构），而且知道了CSS样式（也称为表示），剩下的就是JavaScript（也称为行为）。如果你只知道结构和表示，当然了，创建一个漂亮的页面是没有问题的，不过它们只是页面而已。用JavaScript增加行为时，你就能创建一种交互式的体验。或者，还可以更棒，你能创建完备的Web应用。准备好了吗？下面就在你的Web工具箱里增加一项最有意思、功能最全的技能：JavaScript和编程！

如果还觉得动力不足，那么还有一个理由，这也是最赚钱的技能！

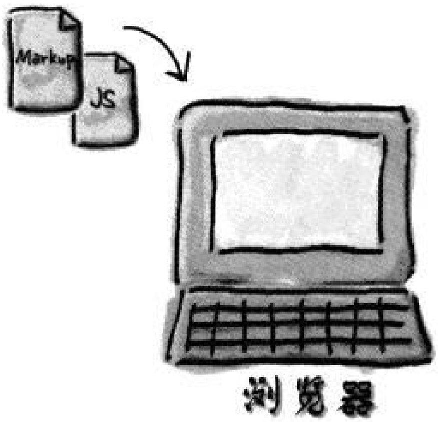
pin51.com 拼吾爱 最新最全大集合网站

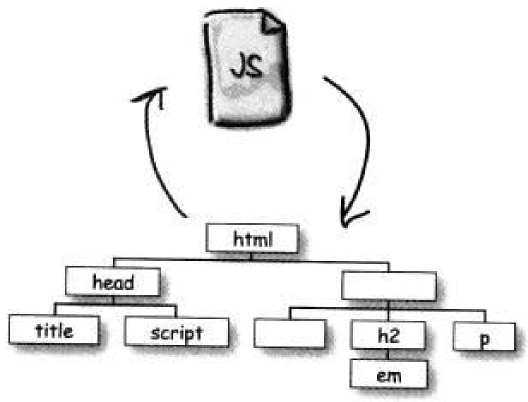
pin51.com 拼吾爱

JavaScript的工作方式

我们的目标是编写JavaScript代码，加载Web页面时在浏览器中运行。这个代码可以对用户的动作做出响应、更新或修改页面、与Web服务通信，总的来讲，可以让页面感觉更像是一个应用而不只是一个文档。下面来看这是如何做到的：

```
<html>
<head>
<script>
 var x = 49;
</script>
<body>
<h1>My first JavaScript</h1>
<p></p>
<script>
 x = x + 2;
</script>
</body>
</html>
```





编写

1

你创建HTML标记和JavaScript代码，并把它们放在文件中，比如说index.html和index.js（或者，也可以都放在HTML文件中）。

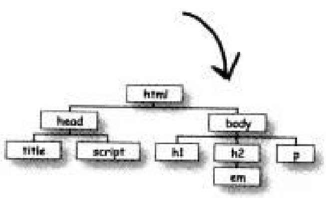
加载

2

浏览器获取并加载你的页面，从上到下解析它的内容。

遇到JavaScript时，浏览器会解析代码，检查它的正确性，然后执行代码。

浏览器还会建立HTML页面的一个内部模型，称为DOM。



运行

3

JavaScript继续执行，使用DOM检查页面、完成修改、从页面接收事件，或者要求浏览器从Web服务器获取其他数据。

用JavaScript能做什么？

一旦有一个包含<script>元素的页面（或者包含引用，指向一个单独的JavaScript文件），你就已经开始编写代码了。JavaScript是一个完备的编程语言，用其他语言能做的事情用JavaScript同样能够做到，甚至还能做得更多，因为我们就在Web页面内部编程！



你可以要求JavaScript：

❶ 建立一个语句

创建一个变量并赋值、让变量相加、完成计算，还可以使用一个JavaScript库的内置功能。

```
var temp = 98.6;
var beanCounter = 4;
var reallyCool = true;
var motto = "I Rule";
temp = (temp - 32) * 5 / 9;
motto = motto + " and so do you!";
var pos = Math.random();
```

❷ 重复做事情

反复地完成语句，次数可以根据你的需要来定。

```
while (beanCounter > 0) {
    processBeans();
    beanCounter = beanCounter - 1;
}
```

❸ 做出判断

编写根据应用的状态并按条件执行的代码。

```
if (isReallyCool) {
    invite = "You're invited!";
} else {
    invite = "Sorry, we're at capacity.";
}
```

声明变量

变量可以存放东西。利用JavaScript，变量可以用来保存各种不同的东西。下面来声明一些变量：

```
var winners = 2;              ← 整数值。
var boilingPt = 212.0;        ← 或浮点数值。
var name = "Dr. Evil";        ← 或者，可以是字符串（我们把它们简称为"串"）。
var isEligible = false;       ← 或者一个布尔值，即true或false。
```

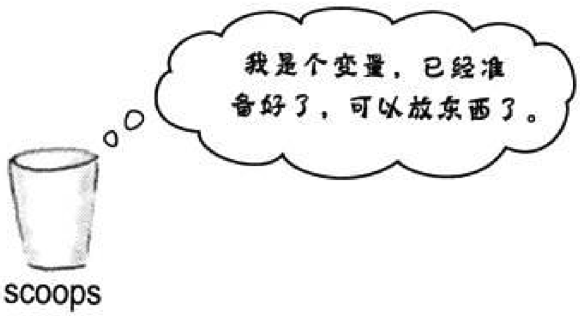


创建变量的3大步骤：

```
 ①              ③  ②
var scoops  =  10;
```

变量是用来存放值的容器。JavaScript变量没有严格的类型，所以任何变量都可以存放数字、串或布尔值。

① 第一步是声明变量，这里就是变量scoops。注意JavaScript与有些语言不同，不需要为变量指定一个类型，它只是创建一个通用容器，其中可以存放多种东西：



② 下一步需要一个值放在这个变量中。可以用多种方式指定一个值：

```
var scoops = 10;                       ← 值可以是一个字面值，如一个数字或一个串。
var scoops = totalScoops / people;     ← 或者，这个值也可以是一个表达式的结果。
var scoops = Math.random() * 10;       ← 还可以使用JavaScript内部库的某个函数，如用一个随机数生成器创建一个值。稍后还会介绍这个函数和你自己的函数。
```

拼吾爱

③ 最后，我们已经有了一个变量，也有了一个值［一个字面值，如10，也可以是计算一个表达式的结果（如totalScoops/people）］，现在要做的就是把这个值赋给这个变量：



当然，一旦创建了一个变量，你可以在任何时刻改变它的值，甚至可以改为一个不同类型的值。下面给出几个例子：

```
scoops = 5;
```
← 可以把scoops重新设置为另一个整数值。

```
scoops = scoops * 10;
```
← 或者甚至可以在改变这个变量值的表达式中使用scoops自身。在这里，scoops将变成50。

```
scoops = "Tired of being an integer";
```
← 另外，可以同时改变scoops的值和类型，在这里会把它改为一个串。要当心，如果你以为scoops是一个数字，这可能会导致代码出现大问题。稍后会介绍这个内容。

```
scoops = null;
```
↑ 或者，JavaScript中还有一个值表示“没有值”，称为null。我们会在后面介绍如何使用这个值。



语法趣事

- 每个语句都以一个分号结束。
  `x = x + 1;`
- 单行注释以两个斜线开头。注释就是为你或其他开发人员提供的代码说明。这些内容不会计算。
  `// I'm a comment`
- 空格没有影响（代码中几乎到处都有空格）。
  `x         =         2233;`
- 字符串用双引号包围。
  `"You rule!"`
- 变量使用var和一个名来声明。不需要类型，这一点与其他一些语言不同。
  `var width;`
- 布尔值true和false两边不要加引号。
  `rockin = true;`
- 声明变量时不一定要指定一个值：
  `var width;`

there are no Dumb Questions

问：如果我这样写，变量的值是什么呢？

```
var winner;
```

答： 执行这个语句之后，变量winner会赋值为undefined，这也是JavaScript的一个特殊值和类型。本书后面会介绍在什么情况下用这个值，以及如何使用。

问：我见过其他编程语言，其中声明变量都要指定一个类型。比如int x或String y。JavaScript没有类型吗？

答： JavaScript当然有类型，不过与你以前用过的语言不同，JavaScript采用的是动态类型，也就是说，你不必指定一个类型，JavaScript解释器会在代码运行时确定要使用什么类型。

pin51.com 拼吾爱

如何命名变量

你可能想知道如何为变量选择名字？如果你以前在HTML元素中指定过元素id，会发现变量与id非常类似。创建变量名只有几条规则。



正式编码

数字、串和布尔在JavaScript中都称为基本类型。变量中还可以存放另外一种东西：对象。很快就会谈到对象，不过现在你可以把对象想成是一些东西的集合，而基本类型只是一个东西，不能再进一步分解了。

规则#1：变量要以一个字母、下划线或者美元符开头。

命名变量时，要想有个好的开端，不仅要有一个有意义的名字，还要使用一个字母（可以是小写或大写）、下划线字符或美元符开头。下面给出几个例子：



```
var thisIsNotAJoke;
var _myVariable;
var $importantVar;
```

要这样……

以数字开头，这样不好。→

用不合法的符号（%和~）开头。→

```
var 3zip;
var %entage;
var ~approx;
```

……不能这样。



规则#2：然后可以使用任意多个字母、数字、下划线或美元符。

继续用字母、美元符和下划线创建变量名。如果愿意，第一个字符后面可以有数字：



```
var my3sons;
var cost$;
var vitaminB12;
```

要这样……

有个空格，这是不允许的。→

有-、+符号。这样不可以，这会让JavaScript不知所措。→

```
var zip code;
var first-name;
var to+do;
```

……不能这样。



拼吾爱

规则#3：一定要避开JavaScript的所有保留字。

JavaScript包含很多保留字，比如if、else、while和for（这只是其中几个例子），另外，如果你试图用这些保留字作为你的变量名，JavaScript可能会不客气。下面给出一个JavaScript保留字列表。现在还不需要你记住这些保留字，在学习JavaScript的过程中你会逐步找到感觉。不过，如果你困惑JavaScript为什么会抱怨说你声明的变量有问题，可以这样想想，“嗯，是不是我用了一个保留字？”

abstract	delete	goto	null	throws
as	do	if	package	transient
boolean	double	implements	private	true
break	else	import	protected	try
byte	enum	in	public	typeof
case	export	instanceof	return	use
catch	extends	int	short	var
char	false	interface	static	void
class	final	is	super	volatile
continue	finally	long	switch	while
const	float	namespace	synchronized	with
debugger	for	native	this	
default	function	new	throw	



要避免使用这些变量名！

there are no Dumb Questions

问：如果我用一个保留字作为变量名的一部分呢？比方说，能不能把一个变量命名为ifOnly（也就是说，变量名中能不能包含保留字if）？

答： 当然可以，只要保证不要与保留字完全匹配就行。而且最好编写清晰的代码，所以一般不会用类似elze的名字，这可能会与else混淆。

问：JavaScript区分大小写吗？换句话说，myvariable和MyVariable一样吗？

答： 如果你熟悉HTML标记，可能会习惯于不区分大小写的语言，毕竟，浏览器会同样看待<head>和<HEAD>。不过，对于JavaScript，大小写是有影响的，myvariable和MyVariable是两个不同的变量。

问：我听说JavaScript任何时候都可以为变量赋值（数字、串等）。不过，如果我让两个变量相加，其中一个是数字，另一个是字符串，会发生什么呢？

答： JavaScript很聪明，它会尝试根据你的需要转换类型。例如，如果你将一个串和一个数字相加，它就会把这个数字转换为一个串，然后将两个串联起来。有些情况下，这很不错，但还有一些情况下这可能并不是你想要的。先想一想，稍后就会讨论这个问题。

Web镇指南：更好的命名



选择变量名时有很大的灵活性，所以我们想给你几个提示，以便你更容易地命名变量：

选择有含义的名字。

像_m、r和foo之类的变量名可能对你来说有某种含义，但是它们在Web镇中太泛滥了。你很可能过一段时间就会忘记它们原来的含义，不仅如此，如果用类似angle、currentPressure和passed等名字，你的代码也会更可读。

创建多词变量名时使用“camel case”记法。

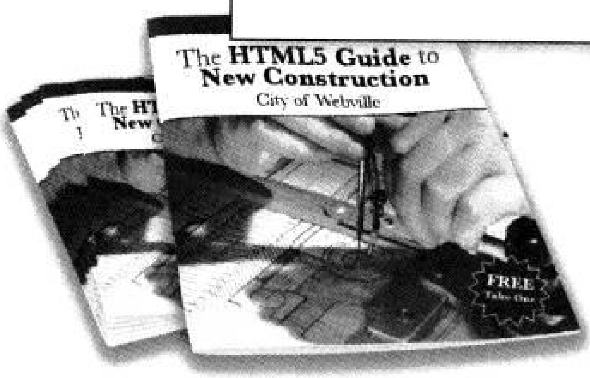
有时候你必须确定如何对一个表示复杂事物的变量命名，比如说，一个喷火的双头龙。怎么做呢？可以使用camel case记法，也就是除了第一个单词外，将每个单词的首字母大写：twoHeadedDragonWithFire。Camel case记法很容易使用，在Web镇中使用也很广泛，可以提供足够的灵活性，能根据你的需要来创建特定的变量名。当然还有其他的方案，不过这种方法更为常用（甚至比JavaScript还流行）。

只有当有充分理由的情况下才使用以_和$开头的变量。

以$开头的变量通常是为JavaScript库保留的，另外有些程序员会由于一些约定使用以_开头的变量，这些变量很少使用，建议你尽量离它们远一点，除非你有非常充分的理由（如果确实有理由你肯定说得出）。

保证安全。

对变量命名时要保证安全。本书后面我们还会给出几个保证安全的提示，不过，对现在来说，只需要记住命名要清楚、避开保留字，另外声明变量时一定要用var。



拼吾爱

需要表达

我们已经见过类似下面的一些JavaScript语句：

一个JavaScript语句

```
scoops = scoops - 1;
```

变量　赋值　表达式

下面来更仔细地研究表达式，比如这个语句中的表达式。实际上，JavaScript中表达式无处不在，所以一定要知道可以表达哪些内容，这很重要。下面给出几种情况……

可以编写得到数字的表达式……

数值表达式

```
(9 / 5) * tempC + 32
x - 1
Math.random() * 10
2.123 + 3.2
```

可以写得到布尔值true或false的表达式（这些显然是布尔表达式）。

布尔表达式

```
2 > 3
startTime > now
tempF < 75
level == 4
pet == "Duck"
```

……还可以写得到串的表达式。

串表达式

```
"super" + "cali" + youKnowTheRest
p.innerHTML
"March" + "21" + "st"
phoneNumber.substring(0, 3)
```

还有另外一些类型的表达式，稍后就会介绍。

其他表达式

```
function () {...}
document.getElementById("pink")
new Array(10)
```

注意下面几页上的表达式（当然也要注意本书后面的部分），你会看到如何用这些表达式在代码中完成计算、重复多次执行，以及做出判断。

Exercise

自我表达！

你已经看到JavaScript中可以使用不同类型的表达式，现在来具体运用这个知识，自己计算几个表达式。本章最后给出了答案。

```
(9 / 5) * tempC + 32
```

tempC为10时结果是什么？ ________

```
"Number" + " " + "2"
```

得到的串是什么？ ________

```
level >= 5
```

level为10时结果是什么？ ________

如果level为5呢？ ________

```
color != "pink"
```

← 提示：!表示非。

如果color是“blue”结果是什么？ ________

```
(2 * Math.PI) * r
```

如果r为3结果是什么？ ________

提示：Math.PI会给出pi的值（你应该知道，就是3.14……）。



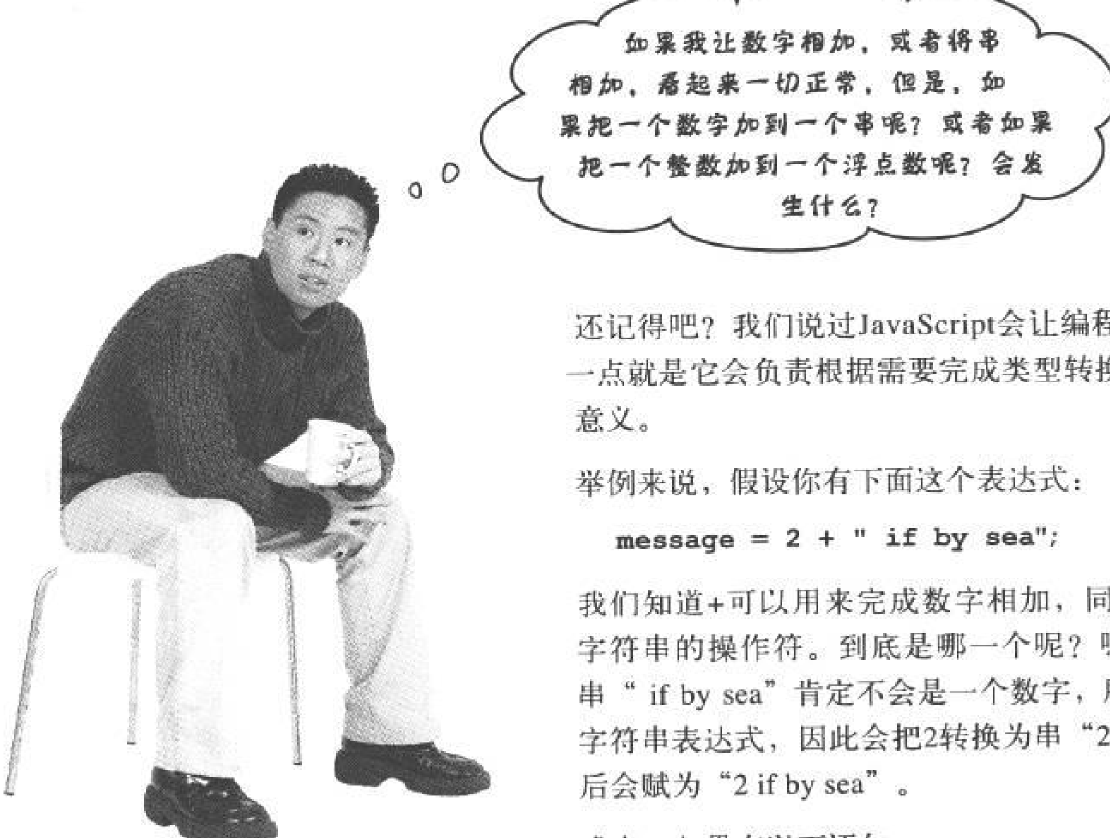
不是这种表达！

Sharpen your pencil

根据你目前对JavaScript变量、表达式和语句的了解，看看你能不能找出以下哪些是合法的，而哪些可能导致一个错误。

在下面的列表中，圈出合法的语句。

```
var x = 1138;
var y = 3/8;
var s = "3-8";
x = y;
var n = 3 - "one";
var t = "one" + "two";
var 3po = true;
var level_ = 11;
var highNoon = false;
var $ = 21.30;
var z = 2000;
var isBig = y > z;
z = z + 1;
z--;
z y;
x = z * t;
while (highNoon) {
    z--;
}
```



还记得吧？我们说过JavaScript会让编程变得很容易。其中一点就是它会负责根据需要完成类型转换，来保证表达式有意义。

举例来说，假设你有下面这个表达式：

```
message = 2 + " if by sea";
```

我们知道+可以用来完成数字相加，同时它也是用来联接字符串的操作符。到底是哪一个呢？嗯，JavaScript知道串“ if by sea”肯定不会是一个数字，所以它认为这是一个字符串表达式，因此会把2转换为串“2”，变量message最后会赋为“2 if by sea”。

或者，如果有以下语句：

```
value = 2 * 3.1;
```

JavaScript会把整数2转换为一个浮点数，所以最后结果是6.2。

不过，可以预见，JavaScript并不总能如你所愿，有些情况下，在转换方面它还需要一点帮助。稍后我们还会讨论这个问题。

BRAIN POWER

JavaScript计算下面的语句时会得到什么？

```
numORString1 = "3" + "4"

numORString2 = "3" * "4"
```

为什么？

pin51.com 拼吾爱

```
while (juggling) {
    keepBallsInAir();
}
```



反反复复……

如果一个JavaScript程序中所有工作都只做一次，这可能是一个很没意思的程序。很多事情都需要做多次。洗头发时，你要冲头，打洗发液，再重复这个过程，直到头发干净为止。再比如开车，你要一直开，直到到达目的地。或者你要不断地用勺舀出冰淇淋，直到全部取光。为了处理这些情况，JavaScript提供了很多途径来循环执行代码块。

可以使用JavaScript的while循环不断地做某件事，直到满足某个条件。

我们有一桶冰淇淋，里面还有10勺。这里声明一个变量，并初始化为10。

while使用一个布尔表达式，计算为true或false。如果为true，就会执行它后面的代码。

只要还剩下多于0勺，就继续做这个代码块中的工作。

```
var scoops = 10;

while (scoops > 0) {
    alert("More icecream!");
    scoops = scoops - 1;
}

alert("life without ice cream isn't the same");
```

每次执行while循环时，会提示用户还有冰淇淋，然后取走1勺，将scoops数减1。

条件(scoops > 0)为false时，循环结束，代码继续从这里执行，开始执行程序的下一行代码。



所以如果考虑使用while循环，首先初始化某个值，比如剩余的冰淇淋勺数，while循环会对这个值完成测试，如果为true，就执行一个代码块。这个代码块将做一些工作，并且会在某个时刻更新条件测试中有关的值，使得条件失败，结束循环。

```
var scoops = 10;

while (scoops > 0) {
    alert("More icecream!");
    scoops = scoops - 1;
}

alert("life without ice cream isn't the same");
```


JavaScript还提供了for循环，使结构更为形式化。下面是用for循环编写的冰淇淋代码：

```
for (scoops = 10; scoops > 0; scoops--) {
    alert("There's more ice cream!");
}
alert("life without ice cream isn't the same");
```


there are no Dumb Questions

问： while和for循环在我看来都一样。分别在什么时候使用呢？

答： 一般来说，用for和while可以做同样的事情。不过，从这个冰淇淋例子可以看到，for循环更为紧凑一些，不过你可能会认为while循环更可读 。所以关键是哪一个最适合你的具体情况。通常，for循环更习惯于对固定数目的值完成迭代（比如，购物车里的商品），而while循环更适合不断循环直至满足某个条件（比如，为用户提供一个测验，直至他做对为止）。

pin51.com 拼吾爱

扮演浏览器



这一页上的每一个JavaScript片段都是一个单独的代码段。你的任务是扮演浏览器，计算各个代码段，回答有关结果的一个问题，并在代码下面写出你的答案。

本章最后会给出答案。

代码片段1

```
var count = 0;
for (var i = 0; i < 5; i++) {
    count = count + i;
}
alert("count is " + count);
```

alert显示的count是多少？

______

代码片段2

```
var tops = 5;
while (tops > 0) {
    for (var spins = 0; spins < 3; spins++) {
        alert("Top is spinning!");
    }
    tops = tops - 1;
}
```

______

你会看到多少次提醒："Top is spinning!"？

代码片段3

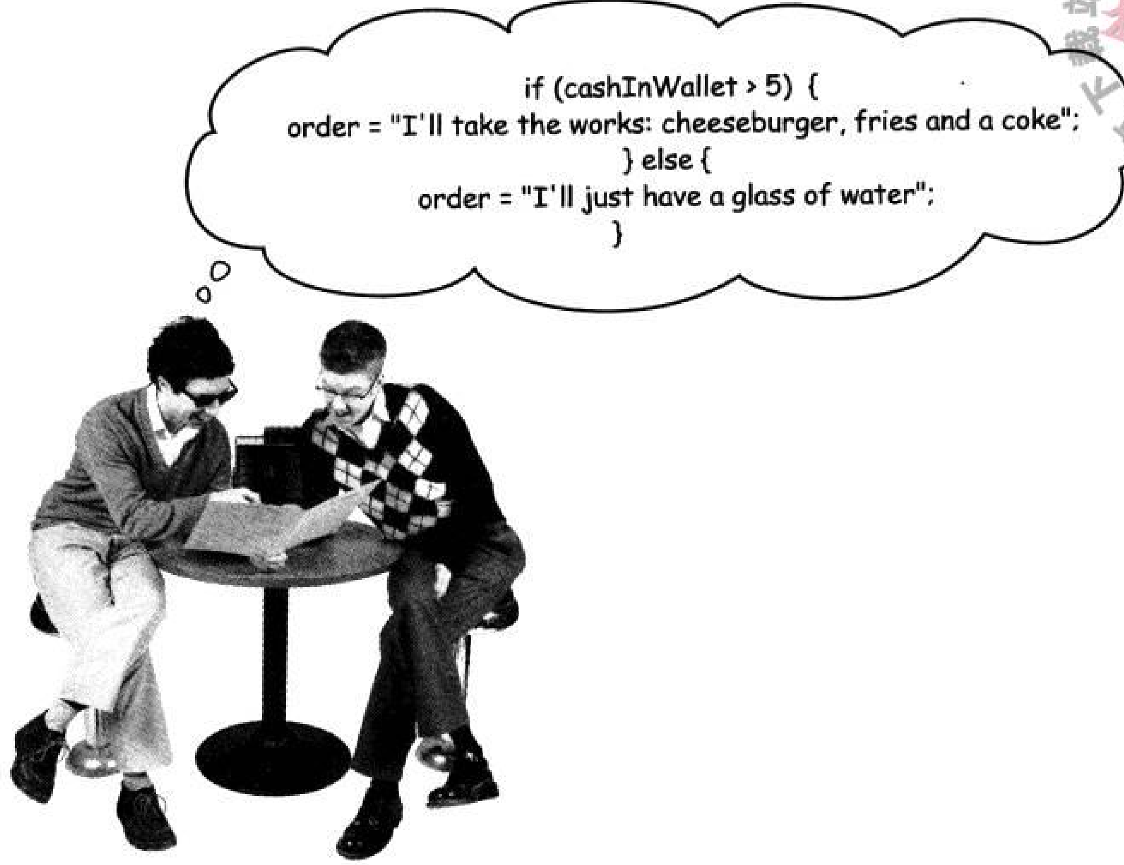
```
for (var berries = 5; berries > 0; berries--) {
    alert("Eating a berry");
}
```

你吃了多少草莓？ → ______

代码片段4

```
for (scoops = 0; scoops < 10; scoop++) {
    alert("There's more ice cream!");
}
alert("life without ice cream isn't the same");
```

______ ← 你吃了多少勺冰淇淋？



用JavaScript做判断

我们已经在for和while语句中使用一个布尔表达式作为条件测试，来确定是否继续循环。JavaScript中还可以用布尔表达式做出判断。下面给出一个例子：

这是我们的布尔表达式，查看还剩下多少勺。

如果少于3勺，就执行代码块。

```
if (scoops < 3) {
   alert("Ice cream is running low!");
}
```

还可以把多个测试串在一起：

可以根据需要用“else if”增加多个测试，每个“else if”都有自己的相关代码块，相应条件为true时则会执行这个代码块。

```
if (scoops < 3) {
   alert("Ice cream is running low!");
} else if (scoops > 9) {
   alert("Eat faster, the ice cream is going to melt!");
}
```

pin51.com

更多判断……另外，增加一个“收容箱”

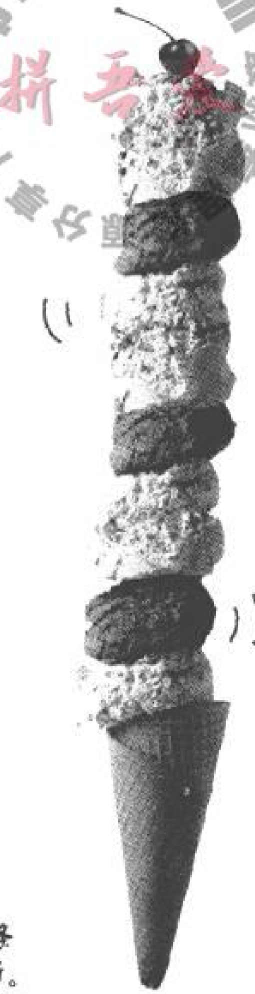
还可以为if语句提供一个无所不包的“收容箱”。也就是最后一个else，所有其他条件都失败时就会运行这个else代码块。下面再增加一些if/else，另外增加一个“收容箱”：

注意这里做了修改，只有当scoops刚好等于3时才发生。

```
if (scoops == 3) {
   alert("Ice cream is running low!");
} else if (scoops > 9) {
   alert("Eat faster, the ice cream is going to melt!");
} else if (scoops == 2) {
   alert("Going once!");
} else if (scoops == 1) {
   alert("Going twice!");
} else if (scoops == 0) {
   alert("Gone!");
} else {
   alert("Still lots of ice cream left, come and get it.");
}
```

我们增加了另外一些条件，倒数到0勺。

这就是我们的“收容箱”。如果上述条件都不为true，这个代码块肯定会执行。





Exercise

把上面的代码插入到下面的while循环中。跟踪执行while循环，按顺序写出得到的提醒。完成后可以查看本章最后给出的答案。

```
var scoops = 10;
while (scoops >= 0) {

    scoops = scoops - 1;
}
alert("life without ice cream isn't the same");
```

在这里插入上面的代码……

把输出写在这里。



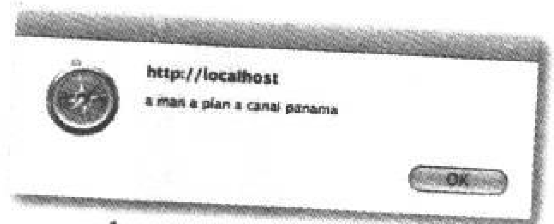
代码磁贴

这个代码在提醒中打印了一个著名的回文。问题是，有些代码写在冰箱磁贴上，可惜掉到地板上了。你的任务是把这些代码捡起来放好，使这个回文代码能正常运行。要当心，地上原来就有一些磁贴，它们不属于这个代码，另外有些磁贴可能要用好几次！学习后面的内容之前，可以检查本章最后给出的答案。

```
var word1 = "a";
var word2 = "nam";
var word3 = "nal p";
var word4 = "lan a c";
var word5 = "a man a p";

var phrase = "";

for (var i = 0; ________; ______) {
    if (i == 0) {
        phrase = ________________;
    }
    else if (i == 1) {
        phrase = ________________ + word4;
    }
    __________ (i == 2) {
        ________________ = phrase + word1 + word3;
    }
    __________ (__________) {
        phrase = phrase + __________ + word2 + word1;
    }
}
alert(phrase);
```



回文就是正着读和反着读都一样的句子！如果你把磁贴都放到了合适的位置，就会看到这个回文。

- else if (i == 0)
- word5
- i == 3
- i++
- else if
- phrase
- i == 4
- word2
- word4
- i < 3
- word1
- i < 4
- else
- word0
- i = 0
- +
- i--
- i = 3
- word3

pin51.com 拼吾爱 最新資源分享下载站



嗯，问得好。首先，你需要知道一些基本知识。到目前为止，我们已经做了不少事情：你知道了如何声明和使用JavaScript变量，而且了解了如何构建基本的语句和表达式。你还知道如何使用所有这些利用`if/else`语句编写条件代码，另外还会用`while`和`for`语句完成迭代工作。

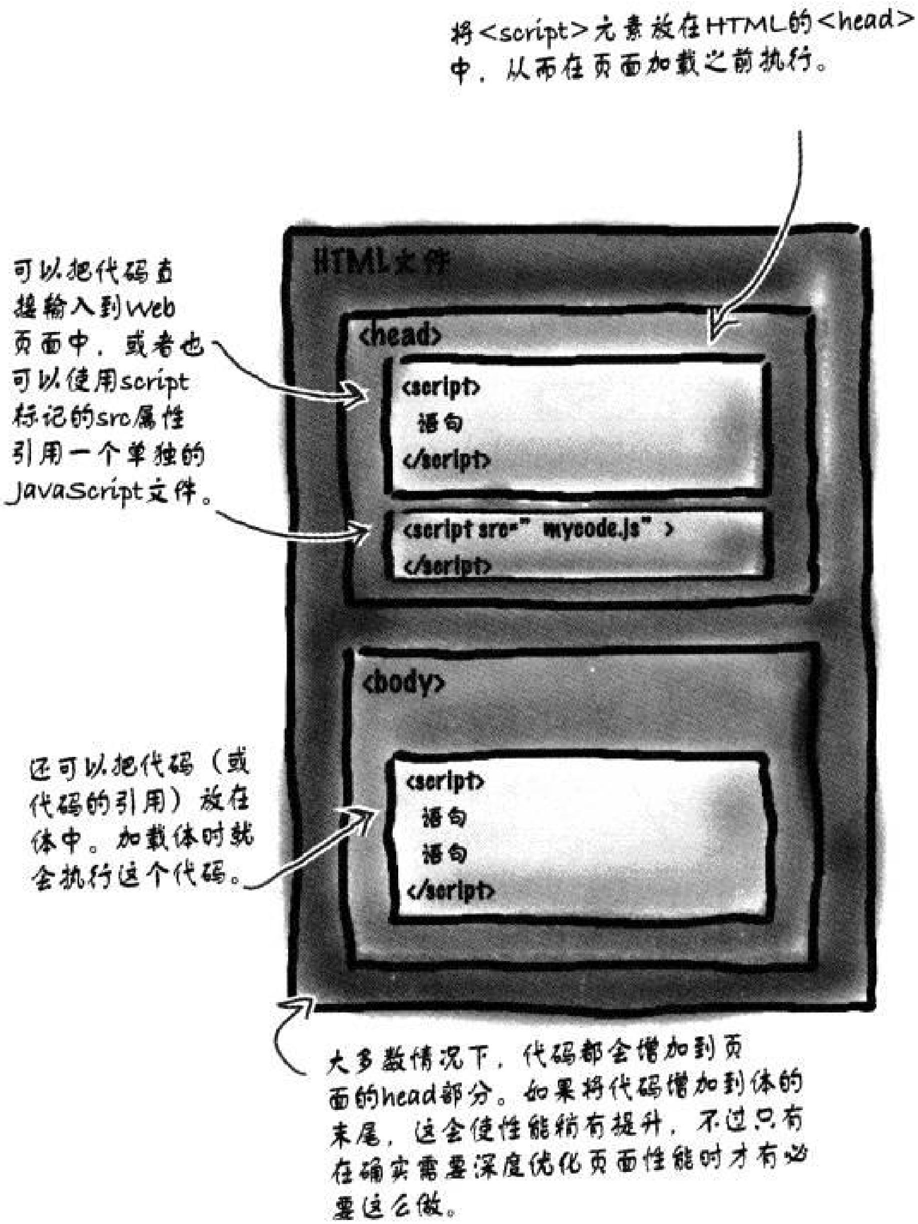
有了这些储备，现在我们来看怎么把JavaScript放在页面中，更重要的是，要知道JavaScript如何与你的页面交互。也就是说，如何确定页面中有些什么，如何改变页面，另外后面还会了解到如何对页面中发生的情况做出反应。

所以，尽管对JavaScript的介绍还没有结束，你也可以不用再等了。现在就来看如何结合标记和行为……

拼吾爱

在页面中增加JavaScript，怎么加？在哪里加？

要使用JavaScript，肯定必须把它增加到一个Web页面中。不过在哪里加呢？另外怎么加？你已经知道有一个<script>元素，所以下面来看在哪里使用这个元素，另外它会如何影响页面中JavaScript的执行。向页面增加代码有3种不同方式：



联机脚本放在<head>元素中。

向页面增加代码时，最常用的方式就是在页面的head部分放置一个<script>元素。在<head>元素中增加JavaScript时，一旦浏览器开始解析head部分就会执行这个代码（这会最先执行），然后才解析页面的其余部分。

通过引用一个单独的JavaScript文件来增加脚本。

还可以链接到一个包含JavaScript代码的单独的文件。将这个文件的URL放在开始<script>标记的src属性中，另外一定要用</script>结束这个脚本元素。如果链接到同一目录中的某个文件，可以只使用该文件名（无需提供完整的路径）。

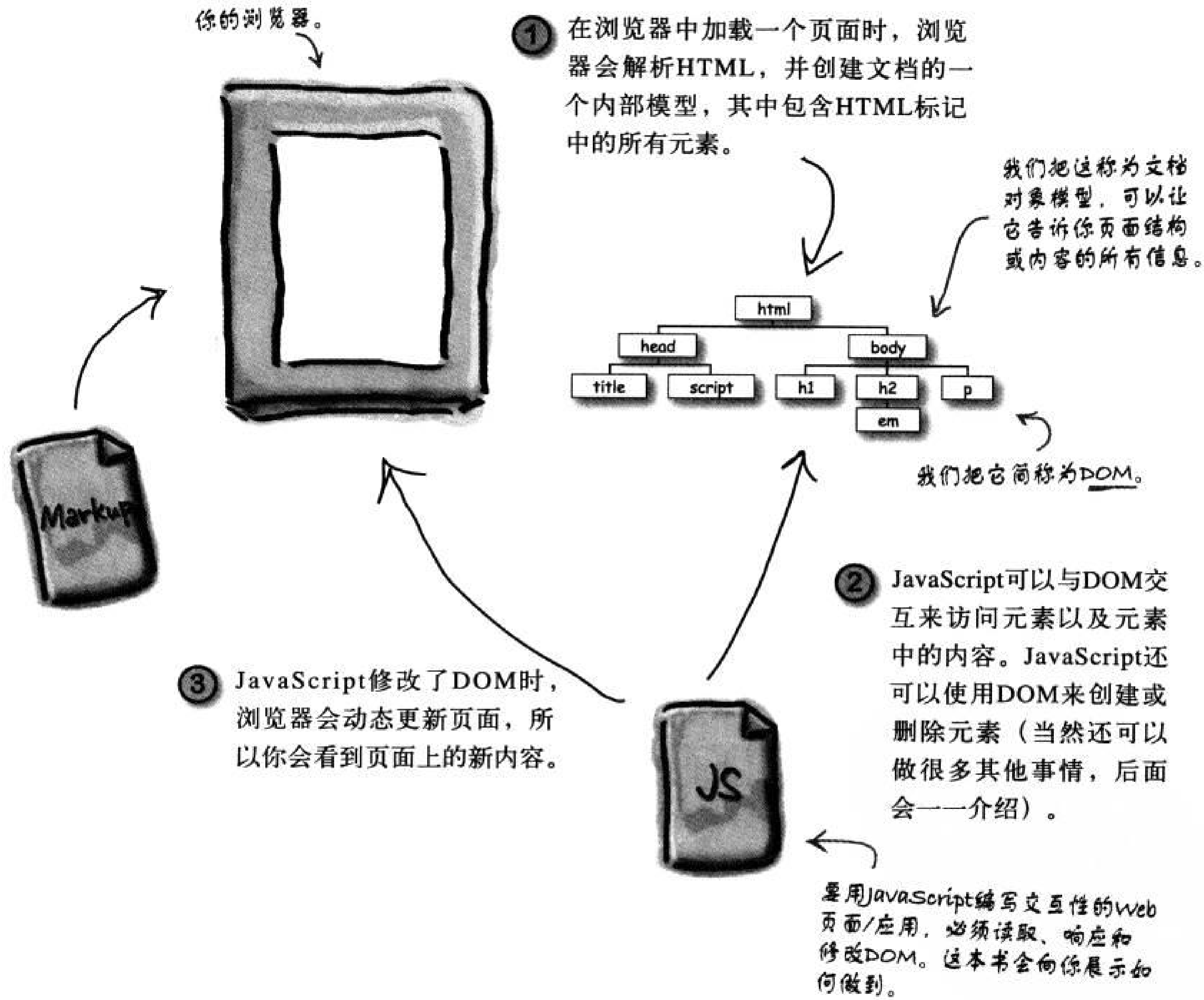
将代码增加到文档体中，可以作为内联代码，也可以作为一个单独文件的链接。

或者，可以把代码直接放在HTML的体中。同样的，将JavaScript代码包围在<script>元素中（或者在src属性中引用一个单独的文件）。浏览器解析体时就会执行页面体中的JavaScript（而且通常会从上向下执行）。

pin51.com 拼吾爱

JavaScript如何与页面交互

JavaScript和HTML是完全不同的两个东西。HTML是标记，而JavaScript是代码。所以怎么让JavaScript与页面中的标记交互呢？答案是可以使用文档对象模型（Document Object Model）。



如何制作你自己的DOM

下面来看一些标记，并为它创建一个DOM。以下是一个简单的菜谱：

配料

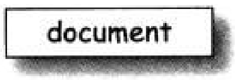
一个格式正确的HTML5页面

一个或多个Web浏览器

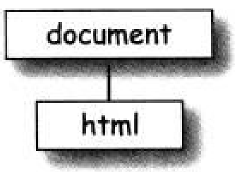
```
<!doctype html>
<html lang="en">
<head>
  <title>My blog</title>
  <meta charset="utf-8">
  <script src="blog.js"></script>
</head>
<body>
  <h1>My blog</h1>
  <div id="entry1">
    <h2>Great day bird watching</h2>
    <p>
      Today I saw three ducks!
      I named them
      Huey, Louie, and Dewey.
    </p>
    <p>
      I took a couple of photos......
    </p>
  </div>
</body>
</html>
```

做法

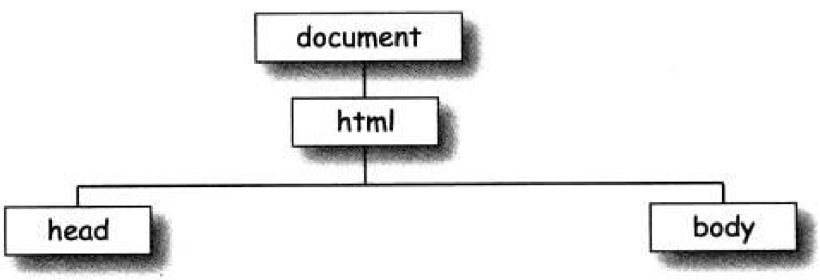
1. 首先在最上面创建一个document节点。



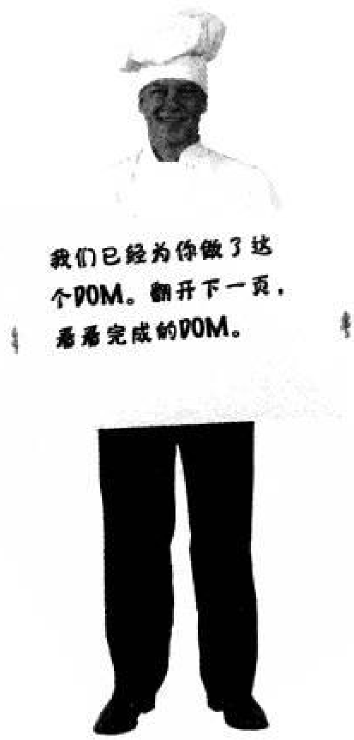
2. 接下来，取HTML页面的最顶层元素，在这里就是<html>元素，称之为当前元素，把它作为document的子节点增加到DOM。



3. 对于当前元素中嵌套的每一个元素，将该元素作为当前元素的子节点增加到DOM。



4. 对于刚增加的各个元素，返回第3步，重复这个工作，直到处理完所有元素。

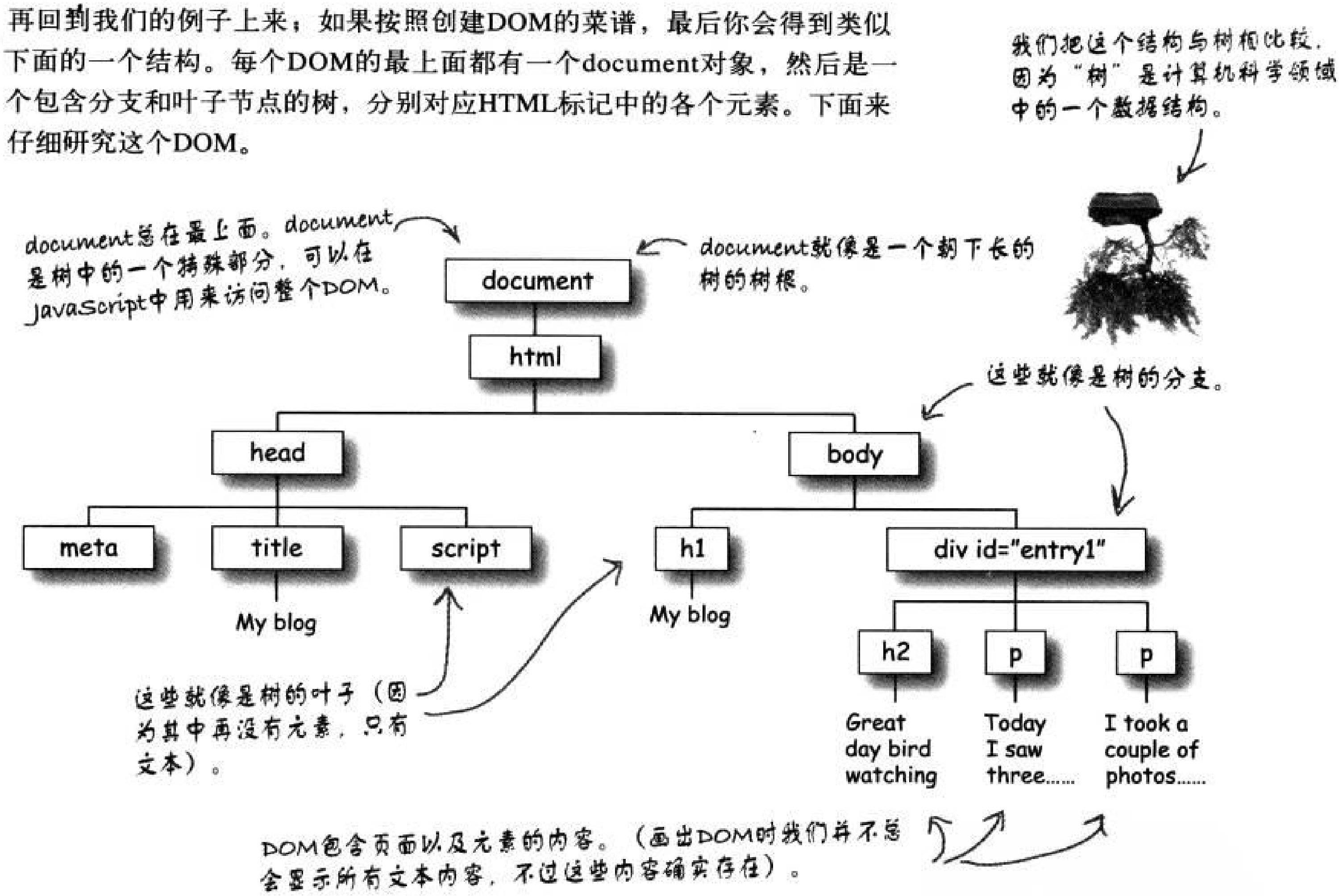


pin51.com 拼吾爱

初尝DOM

文档对象模型（Document Object Model）的妙处在于：它能够在所有浏览器上提供一种一致的方式，通过代码访问HTML的结构和内容。这一点太棒了。稍后就会看到这是如何做到的……

再回到我们的例子上来；如果按照创建DOM的菜谱，最后你会得到类似下面的一个结构。每个DOM的最上面都有一个document对象，然后是一个包含分支和叶子节点的树，分别对应HTML标记中的各个元素。下面来仔细研究这个DOM。



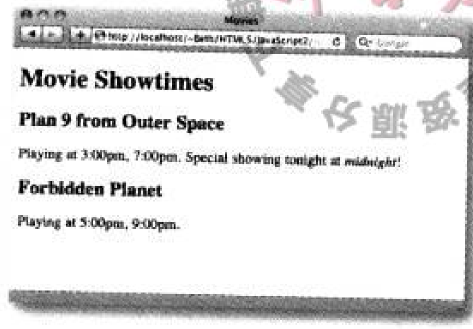




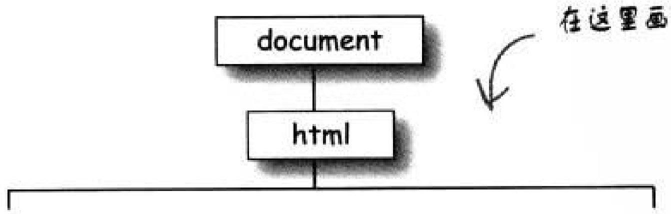
扮演浏览器

你的任务是装扮成浏览器。你需要解析HTML，并由它构建你自己的DOM。请解析右边的HTML，在下面画出DOM。我们已经帮你开了个头。

学习后面的内容之前，请对照本章最后给出的答案检查你做得对不对。

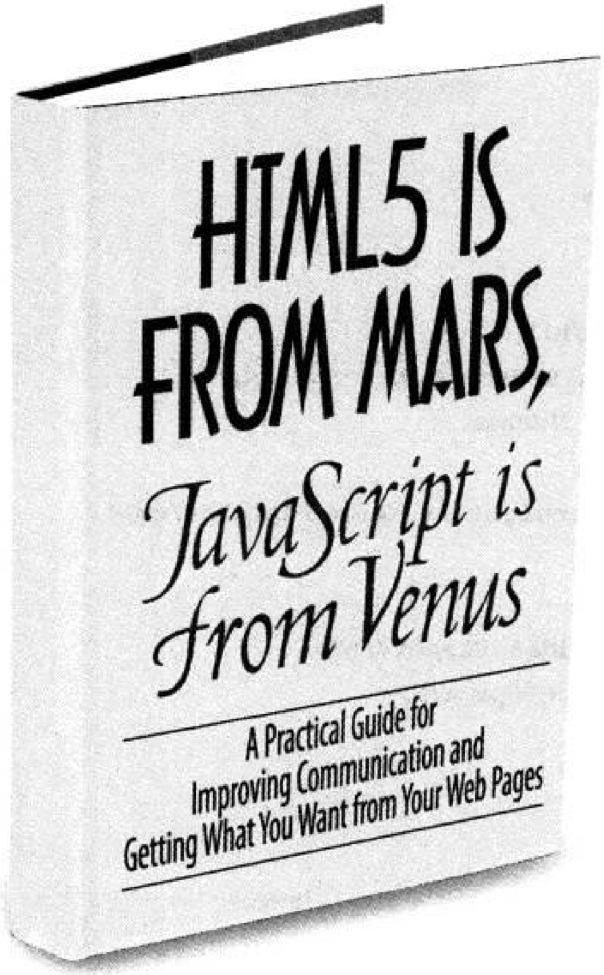



```
<!doctype html>
<html lang="en">
  <head>
    <title>Movies</title>
  </head>
  <body>
    <h1>Movie Showtimes</h1>
    <h2 id="movie1" >Plan 9 from Outer Space</h2>
    <p>Playing at 3:00pm, 7:00pm.
      <span>
        Special showing tonight at <em>midnight</em>!
      </span>
    </p>
    <h2 id="movie2">Forbidden Planet</h2>
    <p>Playing at 5:00pm, 9:00pm.</p>
  </body>
</html>
```



在这里画出你的DOM。

pin51.com 拼吾爱 最新最全资源分享下载站



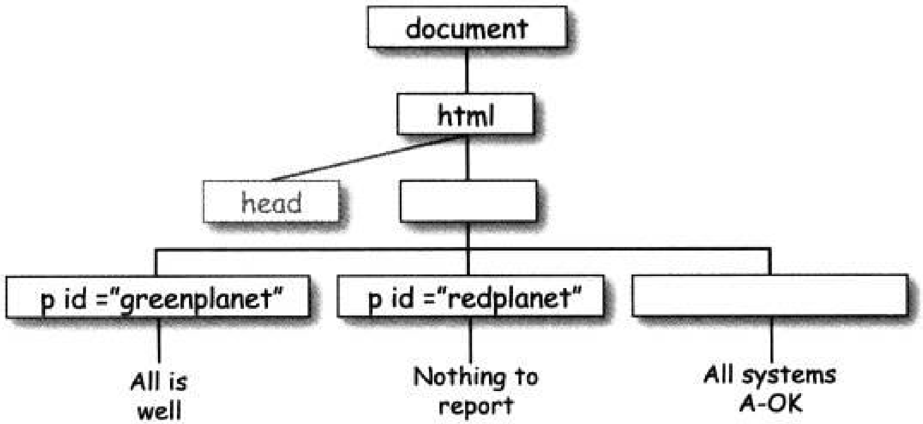
也可以这么说，这两个完全不同的技术如何关联。

HTML和JavaScript完全来自不同的星球。为什么这么讲呢？HTML的DNA由描述性标记构成，允许你描述构成页面的一组嵌套元素。另一方面，JavaScript却是由纯算法性的基因物质构成的,主要用来描述计算。

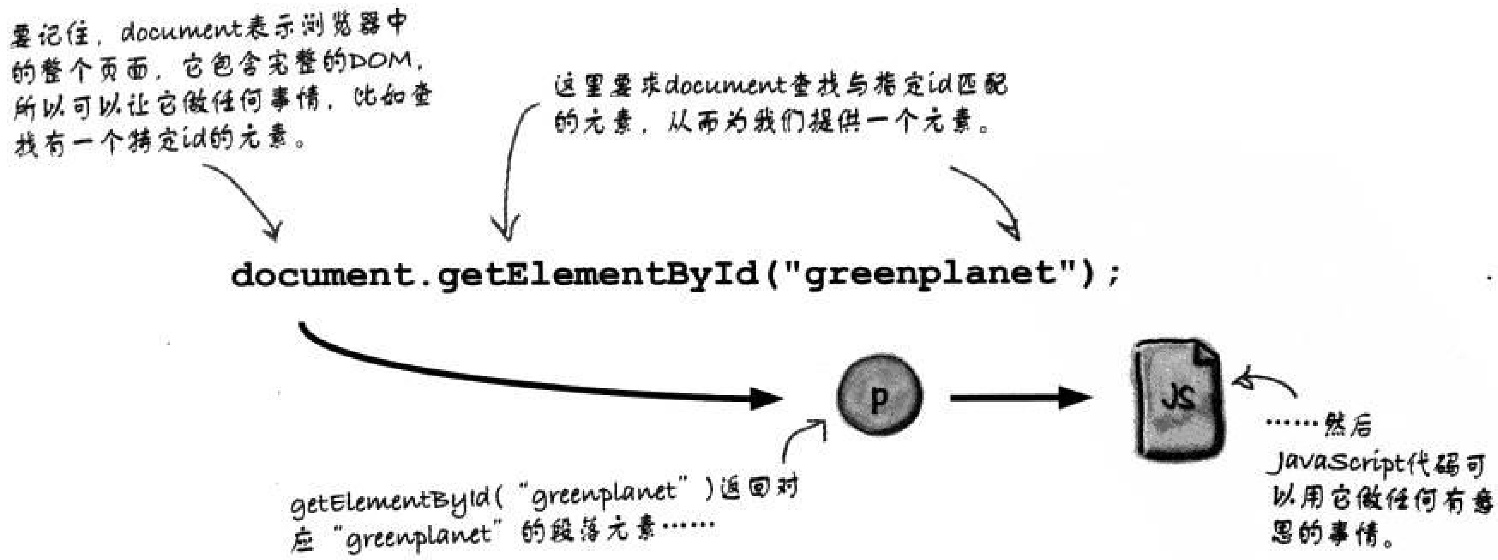
既然它们有着天壤之别，是不是根本无法交流呢？当然不是，因为它们还是有共同之处：DOM。通过DOM，JavaScript就能与页面通信，反之亦然。完成这种通信有很多不同方式，不过现在我们只关心其中的一种。这就像一个小通道，允许JavaScript访问任何元素，这个方法称为getElementById。

下面来看它如何工作……

先从DOM开始。 下面是一个简单的DOM。这里有几个HTML段落，每个段落分别有一个id，分别标识它是绿星球、红星球还是蓝星球。每个段落还分别有一些文本。当然这里还有一个<head>元素，不过，为简单起见，这里省略了有关细节。



下面使用JavaScript做些更有意思的事情。 假设我们想把greenplanet的文本从“All is well”改为“Red Alert: hit by phaser fire!”。以后你可能还希望根据用户的动作（或者甚至根据来自一个Web服务的数据）做这样的工作。不过这在后面再做讨论，现在只是直接更新greenplanet的文本。为此，我们需要得到id为greenplanet的元素。相应的代码如下：



pin51.com 拼吾爱

一旦getElementById提供了一个元素，就可以用它来做些处理（比如把它的文本改为“Red Alert: hit by phaser fire”）。为此，我们通常会把元素赋至一个变量，以便在代码中引用这个元素。下面就这样做，然后修改文本：

将元素赋至一个名为planet的变量。

这里是getElementById调用，它会找出“greenplanet”元素，并返回这个元素。

```
var planet = document.getElementById("greenplanet");
```

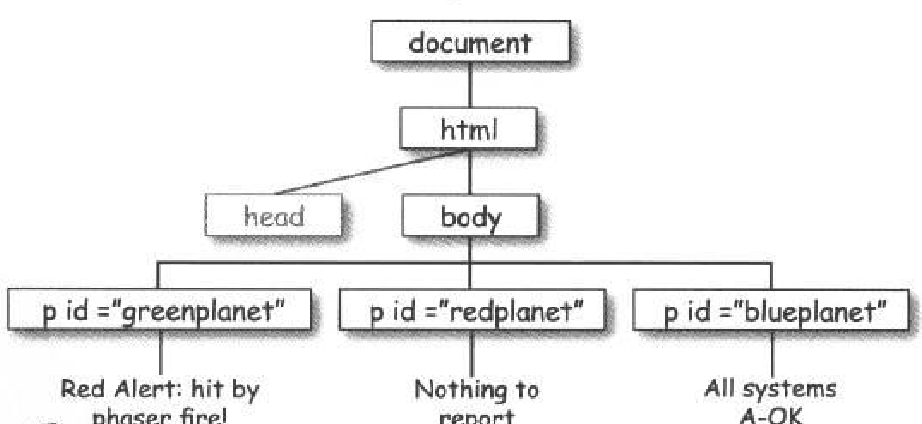
在代码中，现在我们可以直接使用变量planet来引用这个元素。

```
planet.innerHTML = "Red Alert: hit by phaser fire!";
```

可以使用planet元素的innerHTML属性改变元素的内容。

稍后还会讨论元素的属性……

将greenplanet元素的内容改为这个新文本……这会导致DOM（以及页面）得到更新，替换为这个新文本。



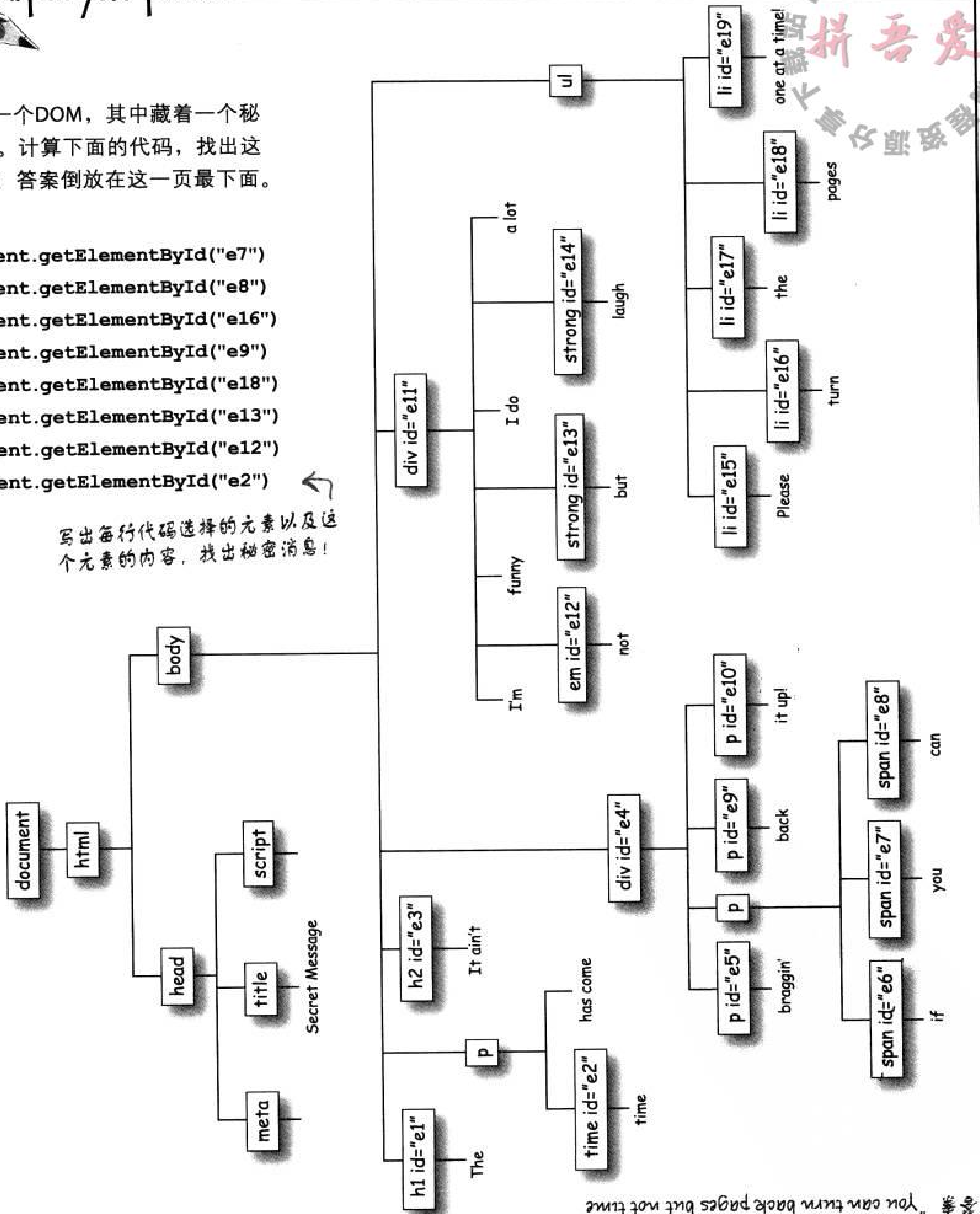
对DOM的任何修改都会反映在浏览器呈现的页面中，所以你会看到这个段落改为包含新的内容！

Sharpen your pencil

下面是一个DOM，其中藏着一个秘密消息。计算下面的代码，找出这个秘密！答案倒放在这一页最下面。

```
document.getElementById("e7")
document.getElementById("e8")
document.getElementById("e16")
document.getElementById("e9")
document.getElementById("e18")
document.getElementById("e13")
document.getElementById("e12")
document.getElementById("e2")
```

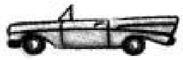
写出每行代码选择的元素以及这个元素的内容，找出秘密消息！



答案："You can turn back pages but not time"

pin51.com 拼吾爱 最新最全 大型站

测试星球





你已经了解如何使用document.getElementById来访问一个元素，也知道了如何使用innerHTML改变这个元素的内容。下面来具体做这个工作，没错，就是现在。

以下给出星球的HTML。head部分有一个<script>元素，我们将在这里放入代码，另外体中包括3个段落，分别对应绿星球、红星球和蓝星球。如果还没有准备好，下面输入HTML以及更新DOM的JavaScript：

```
<!doctype html>
<html lang="en">
<head>
  <title>Planets</title>
  <meta charset="utf-8">
  <script>
    var planet = document.getElementById("greenplanet");
    planet.innerHTML = "Red Alert: hit by phaser fire!";
  </script>
</head>
<body>
  <h1>Green Planet</h1>
  <p id="greenplanet">All is well</p>
  <h1>Red Planet</h1>
  <p id="redplanet">Nothing to report</p>
  <h1>Blue Planet</h1>
  <p id="blueplanet">All systems A-OK</p>
</body>
</html>
```

我们把JavaScript增加到页面的head部分。

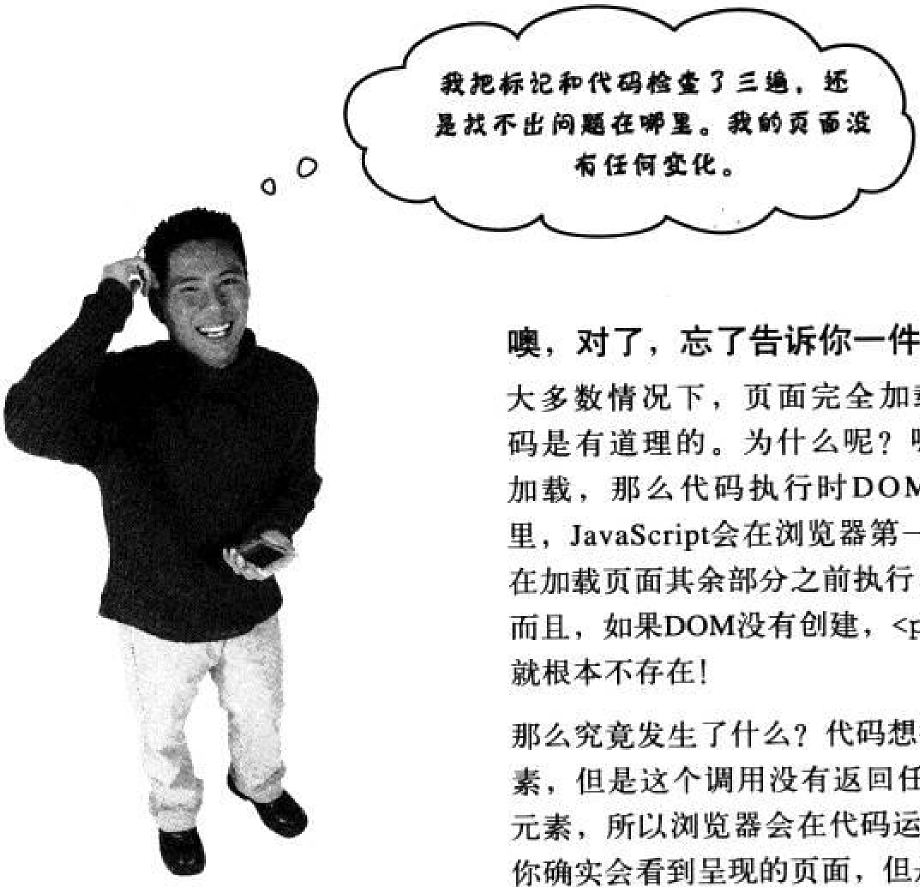
就像前面看到的，这里要得到id为“greenplanet”的<p>元素，并改变它的内容。

这就是要用JavaScript修改的<p>元素。

输入HTML和代码之后，再在浏览器中加载这个页面，看看绿星球上发生的DOM魔法。



唉呀！有问题，绿星球还是显示“All is well”。怎么回事？



噢，对了，忘了告诉你一件事。

大多数情况下，页面完全加载之后再执行JavaScript代码是有道理的。为什么呢？嗯，如果不想等到页面完全加载，那么代码执行时DOM还没有完全创建呢！在这里，JavaScript会在浏览器第一次加载页面的head部分并且在加载页面其余部分之前执行，所以DOM还没有完全创建。而且，如果DOM没有创建，`<p id="greenplanet">`元素就根本不存在！

那么究竟发生了什么？代码想得到id为`greenplanet`的元素，但是这个调用没有返回任何结果，因为并没有匹配的元素，所以浏览器会在代码运行之后继续呈现页面。因此，你确实会看到呈现的页面，但是代码并未改变绿星球中的文本。

我们需要一种方法告诉浏览器“完全加载页面并创建DOM之后再运行我的代码”。下面来看如何做到。

pin51.com 拼吾爱

页面完全加载之前不要打扰DOM

但是如何告诉浏览器只在页面加载之后才执行代码呢？

要告诉浏览器执行代码之前需要等待，我们要用到两部分JavaScript，之前你可能见得不多：一个是window对象，另外还有一个函数。后面还会详细讨论这两个内容，不过现在先直接使用，让代码能正常工作。

将JavaScript代码更新如下：

首先，创建一个名为init的函数，把现有的代码放在这个函数中。

```
<script>
function init() {
    var planet = document.getElementById("greenplanet");
    planet.innerHTML = "Red Alert: hit by phaser fire!";
}

window.onload = init;
</script>
```

注意你的代码放在一个开始{和一个结束}之间。

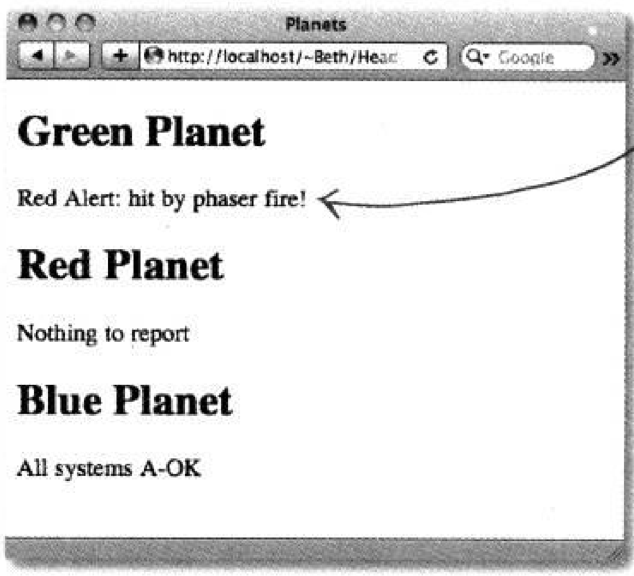
这里将window.onload属性的值设置为这个函数名。

这表示，页面完全加载时要执行init中的代码。

重新加载页面



再来重新加载页面，看看做对了没有。



对了！现在可以看到绿星球<p>元素中的新内容。是不是很棒？

嗯，真正棒的是现在你知道了如何告诉浏览器运行访问元素的代码之前先要等待DOM完全加载。

9in51.com 最新拼吾爱

Sharpen your pencil

以下是一个歌曲播放列表的HTML，只不过这个列表为空。你的任务是完成下面的JavaScript，向这个列表增加歌曲。用合适的JavaScript填空。学习后面的内容之前，先对照本章最后给出的答案检查你做得对不对。

以下是这个页面的HTML。

这是我们的脚本。这个代码要填充下面<ul>中的歌曲列表。

填入缺少的代码，来填充播放列表。

```
<!doctype html>
<html lang="en">
<head>
  <title>My Playlist</title>
  <meta charset="utf-8">
  <script>
  ____________ addSongs() {
    var song1 = document.____________("________");
    var _____ = ____________________("________");
    var _____ = ________.getElementById("________");

    ________.innerHTML = "Blue Suede Strings, by Elvis Pagely";
    __________________ = "Great Objects on Fire, by Jerry JSON Lewis";
    song3.____________ = "I Code the Line, by Johnny JavaScript";
  }
  window.__________ = ____________;
  </script>
</head>
<body>
  <h1>My awesome playlist</h1>
  <ul id="playlist">
    <li id="song1"></li>
    <li id="song2"></li>
    <li id="song3"></li>
  </ul>
</body>
</html>
```

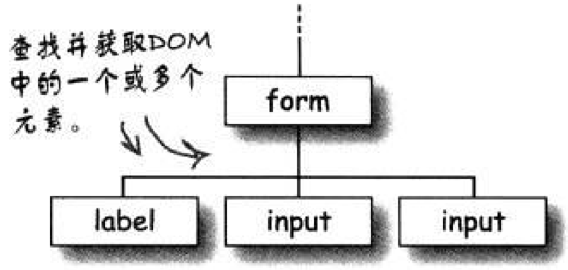
这是空的歌曲列表。上面的代码要向播放列表playlist中的各个<li>增加内容。

如果JavaScript能正常运行，加载页面后会看到类似这样的结果。



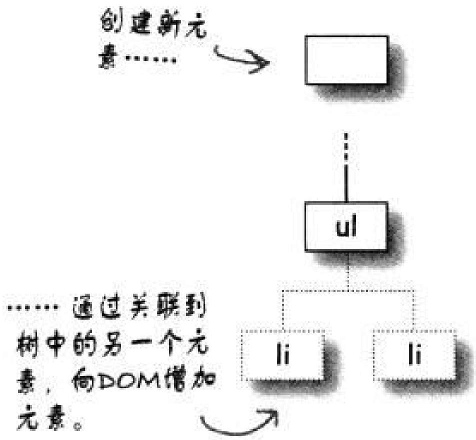
那么，DOM还能做什么？

DOM能做的远不只是我们目前为止所看到的，在这本书中，我们还会用到它的很多其他功能，不过对于现在来说，先简单了解一下，让你有些初步的印象：



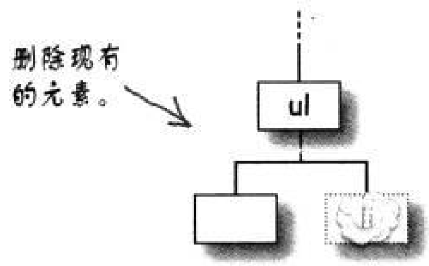
从DOM得到元素。

你当然已经知道这一点，因为我们一直在用document.getElementById，不过还有其他方法来得到元素。实际上，你可以使用标记名、类名和属性来获取不只一个元素，而是整个一组元素（比如类“on_sale”中的所有元素）。还可以得到用户输入的表单值，如一个输入元素的文本。



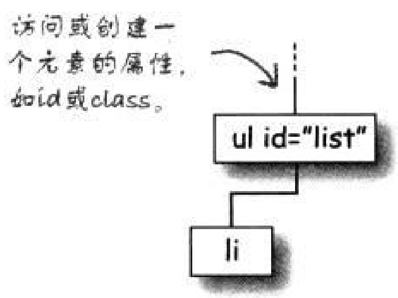
向DOM创建或增加元素。

可以创建新元素，还可以把这些元素增加到DOM。当然，对DOM所做的任何修改都会在浏览器呈现DOM时立即反映出来（这是一件好事）。



从DOM删除元素。

还可以从DOM删除元素，只需取一个父元素，删除它的某个子元素。重申一次，一旦元素从DOM删除，你就会在浏览器窗口中看到它已被删除。



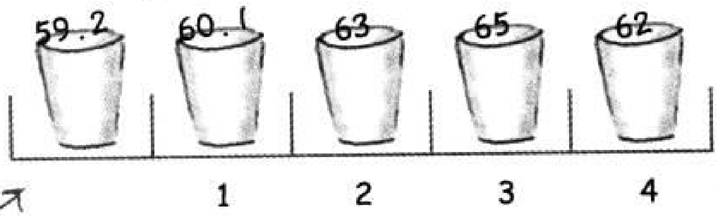
获取和设置元素的属性。

到目前为止，你只是访问了元素的文本内容，实际上还可以访问属性。例如，你可能想知道一个元素的类是什么，然后动态地改变它所属的类。

拼吾爱

能不能再谈谈JavaScript？或者，能不能告诉我JavaScript中如何存储多个值？

你已经对JavaScript和DOM有了一定的了解，可以放松放松，休息一下了，不过在此之前，还想再告诉你另外一个JavaScript类型，以后你会大量使用这个类型——Array。假设你想存储32个冰淇淋口味的名字，或者想存储用户购物车中所有商品的数量，也可能想存储每个小时的室外温度。如果用变量来存储，很快就会变得相当麻烦，特别是如果需要存储数十个、数百个或者成千上万个值时。幸运的是，我们可以让Array来帮忙。



如何创建数组

使用数组之前首先需要创建，而且需要把数组本身赋至一个变量，以便在代码中引用。下面创建以上这个数组，其中包含每个小时的温度：

这是保存数组的变量……

……在这里具体创建一个新的空数组。

```
var tempByHour = new Array();
tempByHour[0] = 59.2;
tempByHour[1] = 60.1;
tempByHour[2] = 63;
tempByHour[3] = 65;
tempByHour[4] = 62;
```

第4章还会再来介绍这个语法，不过对现在来讲，只需要知道它会创建一个新数组。

要向数组增加新值，只需引用数组元素的索引号，并指定一个值。

类似于JavaScript中的变量，可以为一个数组索引赋任何值（或者任何值类型）。

索引。

或者，如果你确实很忙，JavaScript还提供了一个快捷方式，可以输入一个数组（我们称之为“字面量数组”），同时完成创建和初始化：

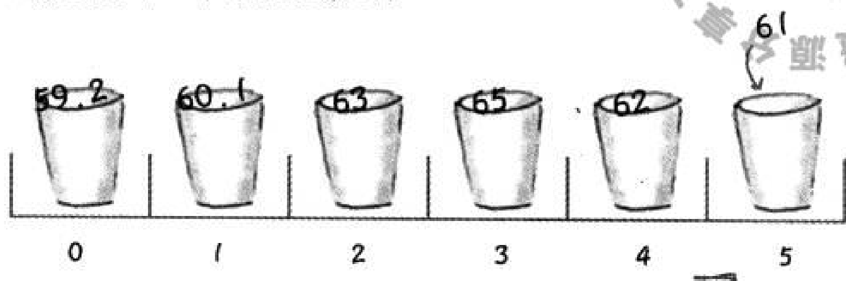
```
var tempByHour = [59.2, 60.1, 63, 65, 62];
```

这也会创建上面同样的数组，但代码少得多。

向数组增加另一个元素

任何时刻都可以不断向数组增加新元素，只需使用下一个未用的索引，如下所示：

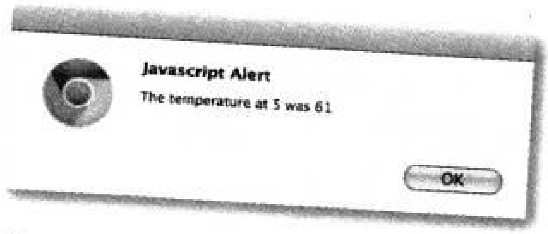
```
tempByHour[5] = 61;
```



通过使用一个新索引，可以得到数组中的一个新元素。

使用数组元素

要得到一个数组元素的值，只需引用数组变量并提供一个索引，如下所示：




```
var message = "The temperature at 5 was " + tempByHour[5];
alert(message);
```

要访问索引为5的温度值，只需引用这个数组并指定索引5。

了解数组大小或其他

只需引用数组的一个属性length，就能很容易地得到数组的大小：

```
var numItems = tempByHour.length;
```

下一章会更深入地讨论属性。现在只要知道每个数组都有这个length数组，可以告诉你数组中的元素个数。

既然我们知道了如何得到一个数组的大小，下面来看能不能把你掌握的循环知识与数组结合起来……

Sharpen your pencil

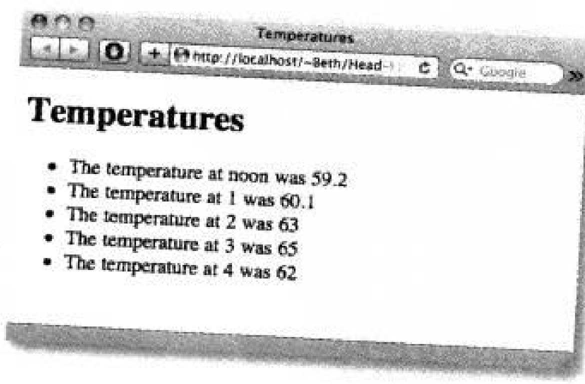
下面的Web页面中包含一个列表，其中元素为空，等着你用JavaScript填入温度。我们已经给出了大部分代码。你的任务是完成这个代码，使它将各个列表项的内容设置为数组中相应的温度（例如，id = “temp0”的列表项会得到数组中索引为0的温度，依此类推）。所以，id = “temp3”的列表项会读做“The temperature at 3 was 65”。如果想加分，看看你能不能想办法得到索引为0的温度，但是读做“noon”而不是0。

```
<!doctype html>
<html lang="en">
<head>
<title>Temperatures</title>
<meta charset="utf-8">
<script>
function showTemps() {
    var tempByHour = new ______________;
    tempByHour[0] = 59.2;
    tempByHour[1] = 60.1;
    tempByHour[2] = 63;
    tempByHour[3] = 65;
    tempByHour[4] = 62;
    for (var i = 0; i < ______________________; ______) {
        var theTemp = _______________[i];
        var id = "____________" + i;
        var li = document.____________________(id);
        if (i == _____) {
            li._______________ = "The temperature at noon was " + theTemp;
        } else {
            li.innerHTML = "The temperature at " + _______ + " was " + ________;
        }
    }
}
window.onload = showTemps;
</script>
</head>
<body>
<h1>Temperatures</h1>
<ul>
    <li id="temp0"></li>
    <li id="temp1"></li>
    <li id="temp2"></li>
    <li id="temp3"></li>
    <li id="temp4"></li>
</ul>
</body>
</html>
```

以下是HTML。

要在这里结合循环和数组。能看出来吗？我们在使用一个可变的索引来访问数组中的各个元素。

上面的代码会把一个包含温度的短语填入各个列表项。





你觉得第1章中的正式商业应用还不算正式，是吧？那好，如果你想展现你的才能，可以试试这一个。

查看这个热门的新Phrase-o-Matic应用的代码，看看能不能搞清楚它的作用……

```
<!doctype html>
<html lang="en">
<head>
    <title>Phrase-o-matic</title>
<meta charset="utf-8">
<style>
body {
    font-family: Verdana, Helvetica, sans-serif;
}
</style>
<script>
function makePhrases() {
    var words1 = ["24/7", "multi-Tier", "30,000 foot", "B-to-B", "win-win"];
    var words2 = ["empowered", "value-added", "oriented", "focused", "aligned"];
    var words3 = ["process", "solution", "tipping-point", "strategy", "vision"];

    var rand1 = Math.floor(Math.random() * words1.length);
    var rand2 = Math.floor(Math.random() * words2.length);
    var rand3 = Math.floor(Math.random() * words3.length);

    var phrase = words1[rand1] + " " + words2[rand2] + " " + words3[rand3];
    var phraseElement = document.getElementById("phrase");
    phraseElement.innerHTML = phrase;
}
window.onload = makePhrases;
</script>
</head>
<body>
    <h1>Phrase-o-Matic says:</h1>
    <p id="phrase"></p>
</body>
</html>
```

拼吾爱 最新最全资源分享下载站 pin5i.com

Phrase-O-Matic

我们希望你会发现这个代码是一个理想的工具，可以帮助你创建下一个热门的启动项目标语。过去，它曾经创建过一些成功的项目口号，比如“双赢增值方案”和“24/7助力过程”，而且我们深信将来它还能用来创建更多漂亮的口号。下面来看具体是如何做的：

① 首先，定义makePhrases函数，页面完全加载后运行这个函数，这样我们就能知道可以安全地访问DOM：

定义一个名为makePhrases的函数，以后会调用这个函数。

```
function makePhrases() {
    
}
window.onload = makePhrases;
```

makePhrases的所有代码都放在这里，稍后来分析……

页面一旦完成加载就运行makePhrases。

② 有了以上的基础，下面可以编写makePhrases函数的代码了。首先建立3个数组，分别包含用来创建短语的单词。我们将使用快捷方式来创建这些数组：

创建一个名为words1的变量，用来引用第一个数组。

```
var words1 = ["24/7", "multi-Tier", "30,000 foot", "B-to-B", "win-win"];
```

在这个数组中放入5个字符串。完全可以把它们换成当前最新的流行语。

```
var words2 = ["empowered", "value-added", "oriented", "focused", "aligned"];
var words3 = ["process", "solution", "tipping-point", "strategy", "vision"];
```

这是另外两个单词数组，分别赋至两个新变量：words2和words3。

pin51.com 拼吾爱 最新最全资源分享下载站

③ 好了，我们已经有了3个新数组存放最新的流行语。现在要做的是随机地从各个数组中选择一个单词，然后把它们放在一起创建一个短语。

如下从各个数组中选择一个单词：

我们为各个数组分别创建一个随机数，将它赋至一个新变量（rand1、rand2和rand3）。

```
var rand1 = Math.floor(Math.random() * words1.length);
var rand2 = Math.floor(Math.random() * words2.length);
var rand3 = Math.floor(Math.random() * words3.length);
```

这个代码会根据各个数组中的元素个数生成一个随机数（在这里是5个元素，不过完全可以向数组增加更多元素，代码一样能正常工作）。

④ 现在把各个随机选择的单词连接在一起，每个单词之间有一个空格来保证可读性，这样就能创建引人注目的宣传语。

用另外一个变量保存这个短语。

使用各个随机数作为单词数组的索引……

```
var phrase = words1[rand1] + " " + words2[rand2] + " " + words3[rand3];
```

⑤ 差不多了，现在有了短语，只需要把它显示出来。你现在应该已经知道该怎么做了：我们要使用getElementById找到段落元素，然后使用元素的innerHTML放入这个新短语。

得到id为“phrase”的<p>元素。

```
var phraseElement = document.getElementById("phrase");
phraseElement.innerHTML = phrase;
```

然后把<p>元素的内容设置为这个短语。

6 好了，完成最后一行代码，再整体检查一遍，在浏览器中加载这个页面之前，先享受享受成就感。测试一下，欣赏得到的宣传语吧。

这是我们看到的结果！



只需重新加载页面，就能看到无穷无尽的启动项目口号（嗯，也许不是无穷无尽，不过暂且认为如此吧。我们还会让这个简单的代码变得更令人兴奋）！

there are no Dumb Questions

问： **到底什么是Math，另外Math.random和Math.floor做什么用？**

答： Math是一个内置的JavaScript库，包含一大堆与数学相关的函数。Math.random会生成0到1之间的一个随机数。我们将这个随机数乘以数组中的元素个数（使用数组的length属性得到），可以得到介于0到数组长度之间的一个数。这个结果很可能是一个浮点数，比如3.2，所以我们使用了Math.floor，来确保得到一个整数，可以把这个整数用做数组的索引，来选择随机的单词。Math.floor所做的就是去除一个浮点数中小数点后面的数字。例如，Math.floor(3.2)就是3。

问： **在哪里可以找到有关Math之类的文档呢？**

答： 关于JavaScript有一个很好的参考资料：O'Reilly出版David Flanagan所著的《JavaScript: The Definitive Guide》。

问： **之前你说过可以在变量中存储基本类型（如数字、串和布尔值），也可以存储对象。不过我们现在是在变量中存储数组。那么数组到底是一个基本类型还是一个对象呢？**

答： 问得好！数组是JavaScript内置的一种特殊类型的对象。之所以说它特殊，这是因为你可以使用数值索引访问数组中存储的值，这是用其他（非数组）对象或者你自己创建的对象做不到的。第4章会介绍如何创建你自己的对象。

问： **如果我试图访问一个不存在的数组索引会怎么样呢？比如说，myWords中存储了5个单词，但我想访问myWords[10]。**

答： 你会得到undefined，如果还记得，这是一个尚未赋值的变量的值。

问： **能从Array删除一个元素吗？如果能，其他元素的索引会有什么变化？**

答： 当然可以从Array删除一个元素，而且有两种不同的方法来删除。可以将该索引的数组元素值设置为null。例如，myArray[2] = null。不过，这意味着Array的长度保持不变。或者还可以完全删除这个元素（使用函数splice）。在这种情况下，所删除元素后面的所有元素的索引都减1。所以，如果myArray[2] = "dog"，myArray[3] = "cat"，而你删除了"dog"，那么就会有myArray[2] = "cat"，而且数组的长度比原来少1。

pin5i.com 拼吾爱

学习一种语言是一个艰巨的工作，不仅要求你开动大脑，还需要让大脑适当放松。所以学完这一章之后，我们需要休息一下，让自己轻松轻松，不过在此之前，先查看后面的要点，并完成填字游戏，真正把这些内容牢牢记住。



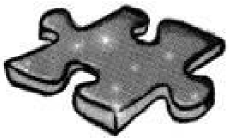
我们还不知道怎么来模拟转换，所以还需要你自助取餐！

www.pin51.com 拼吾爱 最新最全下载站

BULLET POINTS

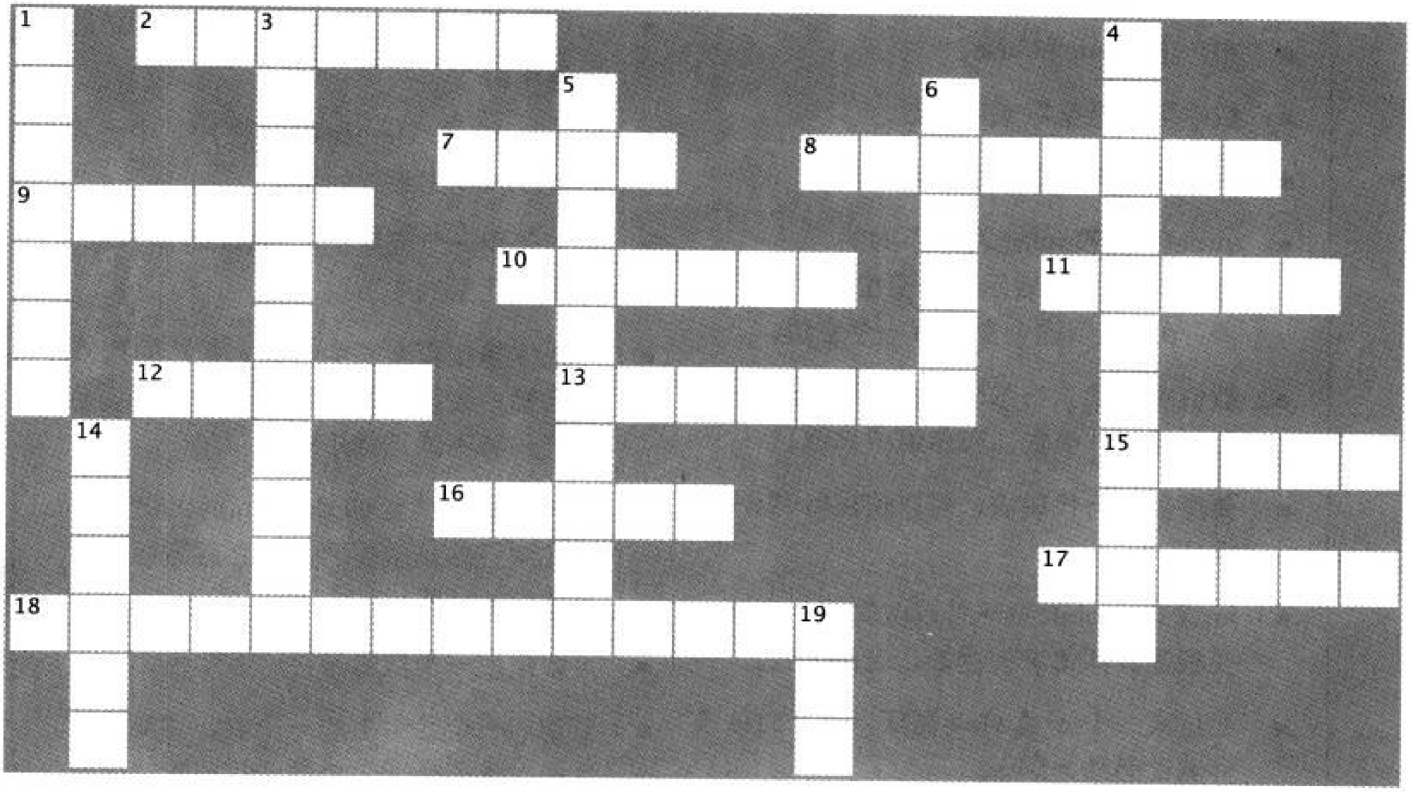
- 使用var声明一个JavaScript变量。
- 数字、布尔值和串是基本类型。
- 布尔值为true和false。
- 数字可以是整数或浮点数。
- 未赋值的变量值为undefined。
- undefined和null是两个不同的值。undefined表示一个变量未赋值；null表示这个变量有一个空值。
- 数值表达式、布尔表达式和串表达式会分别得到一个数、布尔值或串值。
- 要重复执行代码块，可以使用for或while循环。
- for循环和while循环可以做同样的事情，要根据具体情况使用最适合的形式。
- 要结束一个for或while循环，某个时刻条件测试必须为false。
- 可以使用if/else语句根据一个条件测试做出判断。
- 条件测试是布尔表达式。
- 可以向Web页面的head部分或体部分增加JavaScript，或者把它放在单独的文件中，并从Web页面链接这个文件。
- 要用<script>元素包围你的JavaScript代码（或指向代码的链接）。
- 浏览器加载一个Web页面时，它会创建一个文档对象模型（Document Object Model，DOM），这是Web页面的一个内部表示。
- 通过使用JavaScript检查和修改DOM，可以使你的Web页面有交互性。
- 可以使用document.getElementById访问页面中的一个元素。
- document.getElementById使用一个元素的id在DOM中查找元素。
- 可以使用一个元素的innerHTML属性修改这个元素的内容。
- 如果你想在页面完全加载之前访问或修改元素，会得到一个JavaScript错误，你的代码将不能正常工作。
- 将一个函数赋至window.onload属性，可以在浏览器完成页面加载之后运行这个函数中的代码。
- 可以使用一个数组来存储多个值。
- 要访问一个数组中的一个值，需要使用索引。索引是一个整数，指定数组中元素的位置（从0开始）。
- 数组的length属性会告诉你数组中有多少个元素。
- 通过结合循环和数组，可以按顺序访问一个数组中的各个元素。
- Math是一个JavaScript库，包含大量与数学相关的函数。
- Math.random会返回一个介于0和1的浮点数（但是不会为1）。
- Math.floor把一个浮点数小数点后面的所有位去除，将它转换为一个整数。

pin51.com 拼吾爱 最新最全资源分享下载站



HTML5填字游戏

下面用一个填字游戏让你的左右脑都动一动。玩得开心！



横向

2. 5 < 10 是一个 ________ 表达式。
7. 可以将JavaScript增加到HTML的________ 部分或体。
8. ____________ 是DOM树的根。
9. 变量以________ 、$或_开头。
10. DOM是________的内部表示。
11. 使用一个 ________从数组得到一个值。
12. 要选择有意义的名字，对长名要使用________ case记法。
13. 如果写为3 + “Stooges”，JavaScript会把3 ________ 为一个串。
15. 将所有冰淇淋口味都保存在一个 ________中。
16. 可以用一个__________循环反复做事情。
17. 检查________就能知道一个数组中有多少个元素。
18. 通过document.__________ 可以用JavaScript从文档中得到一个元素。

纵向

1. while和for循环使用一个__________表达式作为条件测试。
3. 浏览器加载一个页面时会构建一个文档________ ________。
4. 短语生成器选择的星球id。
5. 增加__________可以使Web页面有交互性。
6. 如果JavaScript在一个HTML页面中，要用一个<__________>标记包围JavaScript代码。
14. 如果快结束了，则可以喝喝茶， ________ 没有结束，那就继续工作！
19. 页面完全加载之前不要打扰________ 。

Exercise Solution

自我表达！

你已经看到JavaScript中可以使用不同类型的表达式。现在来具体运用这个知识，自己计算几个表达式。下面是我们的答案。

```
(9 / 5) * tempC + 32
```

tempC为10时结果是什么？ 50

```
"Number" + " " + "2"
```

得到的串是什么？ Number 2

```
level >= 5
```

level为10时结果是什么？ true

如果level为5呢？ true

>= 是“大于或等于”

```
color != "pink"
```

color “不等于” pink

如果color是“blue”结果是什么？ true

```
(2 * Math.PI) * r
```

如果r为3结果是什么？ 18.84

近似！

提示：Math.PI会给出pi的值（你应该知道，就是3.14……）。



不是这种表达！

Sharpen your pencil Solution

根据你目前对JavaScript变量、表达式和语句的了解，看看你能不能找出以下哪些是合法的，而哪些可能导致一个错误。

在下面的列表中，圈出合法的语句。

```
var x = 1138;
var y = 3/8;
var s = "3-8";
x = y;
var n = 3 - "one";
```

理论上讲，这是合法的，不过会得到一个你无法使用的值。

```
var t = "one" + "two";
var 3po = true;
```

不合法！

```
var level_ = 11;
var highNoon = false;
var $ = 21.30;
var z = 2000;
var isBig = y > z;
z = z + 1;
z--;
z y;
```

不合法！

```
x = z * t;
while (highNoon) {
    z--;
}
```

pin51.com 拼吾爱 最新最全 资源共享 诚信安全

扮演浏览器答案



这一页上的每一个JavaScript片段都是一个单独的代码段。你的任务是扮演浏览器，计算各个代码段，回答有关结果的一个问题，并在代码下面写出你的答案。

代码片段1

```
var count = 0;
for (var i = 0; i < 5; i++) {
    count = count + i;
}
alert("count is " + count);
```

alert显示的count是多少？

10

每次循环迭代时，会把i的值增加到count，而且i会递增，所以每次循环时并不是加1，而是会分别加0、1、2、3和4。

代码片段2

```
var tops = 5;
while (tops > 0) {
    for (var spins = 0; spins < 3; spins++) {
        alert("Top is spinning!");
    }
    tops = tops - 1;
}
```

外部while循环运行5次，内部for循环在每个外循环中运行3次，所以总数为5 x 3，也就是15！

15

你会看到多少次提醒："Top is spinning!"

代码片段3

在这里，我们从5开始循环，直到berries为0，每次berries会递减（而不是递增）。

```
for (var berries = 5; berries > 0; berries--) {
    alert("Eating a berry");
}
```

你吃了多少草莓？ 5

代码片段4

```
for (scoops = 0; scoops < 10; scoop++) {
    alert("There's more ice cream!");
}
alert("life without ice cream isn't the same");
```

这个容易。我们循环了10次，所以吃了10勺！

10 你吃了多少勺冰淇淋？

pin5i.com 拼吾爱 最新最全专业资源分享下载站



Exercise Solution

把上面的代码插入到下面的while循环中。跟踪执行while循环，按顺序写出得到的提醒。以下是我们的答案。

```
var scoops = 10;
while (scoops >= 0) {
    if (scoops == 3) {
        alert("Ice cream is running low!");
    } else if (scoops > 9) {
        alert("Eat faster, the ice cream is going to melt!");
    } else if (scoops == 2) {
        alert("Going once!");
    } else if (scoops == 1) {
        alert("Going twice!");
    } else if (scoops == 0) {
        alert("Gone!");
    } else {
        alert("Still lots of ice cream left, come and get it.");
    }
   scoops = scoops - 1;
}
alert("Life without ice cream isn't the same.");
```

插入的代码。

scoops为3时，这会发生一次。

这也发生一次（scoops为10时）。

这些都只发生一次，条件是scoops分别为2、1和0。

所有其他条件都不为true时才会发生，也就是说，scoops为9、8、7、6、5和4时会执行这个代码块。

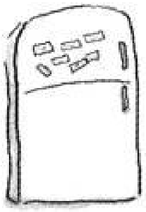
每次循环时将scoops减1。

这会在循环结束后运行。

得到的提醒：

```
Eat faster, the ice cream is going to melt!
Still lots of ice cream left, come and get it.
Still lots of ice cream left, come and get it.
Still lots of ice cream left, come and get it.
Still lots of ice cream left, come and get it.
Still lots of ice cream left, come and get it.
Still lots of ice cream left, come and get it.
Ice cream is running low!
Going once!
Going twice!
Gone!
Life without ice cream isn't the same.
```

pin51.com 拼吾爱



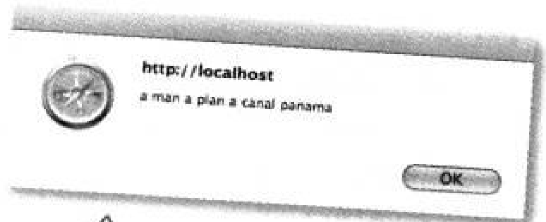
代码磁贴答案

这个代码在提醒中打印了一个著名的回文。问题是，有些代码写在冰箱磁贴上，可惜掉到地板上了。你的任务是把这些代码捡起来放好，使这个回文代码能正常运行。要当心，我们放了一些多余的磁贴，另外有些磁贴可能要用好几次！下面是我们的答案。

```
var word1 = "a";
var word2 = "nam";
var word3 = "nal p";
var word4 = "lan a c";
var word5 = "a man a p";

var phrase = "";

for (var i = 0; i < 4 ; i++ ) {
    if (i == 0) {
        phrase = word5 ;
    }
    else if (i == 1) {
        phrase = phrase + word4;
    }
    else if (i == 2) {
        phrase = phrase + word1 + word3;
    }
    else if ( i == 3 ) {
        phrase = phrase + word1 + word2 + word1;
    }
}
alert(phrase);
```



回文就是正着读和反着读都一样的句子！如果你把磁贴都放到了合适的位置，就会看到这个回文。

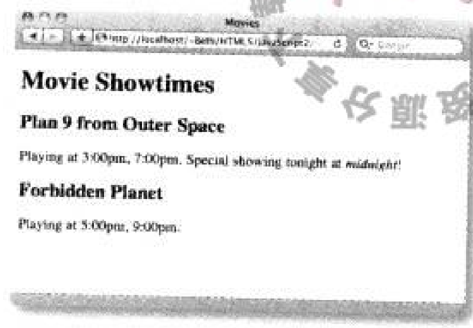
多余的磁贴。

`else if (i == 0)` `i == 4` `word2` `word4` `i < 3` `else` `word0` `i = 0` `+` `i--` `i = 3` `word3`

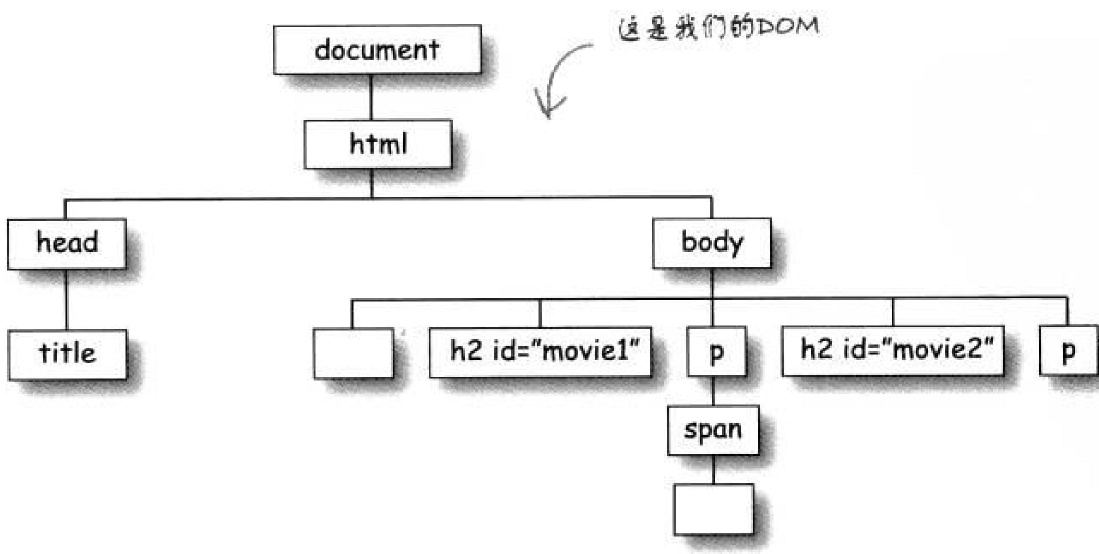


扮演浏览器答案

你的任务是装扮成浏览器。你需要解析HTML，并由它构建你自己的DOM。请解析右边的HTML，在下面画出DOM。我们已经帮你开了个头。




```
<!doctype html>
<html lang="en">
  <head>
    <title>Movies</title>
  </head>
  <body>
    <h1>Movie Showtimes</h1>
    <h2 id="movie1" >Plan 9 from Outer Space</h2>
    <p>Playing at 3:00pm, 7:00pm.
      <span>
        Special showing tonight at <em>midnight</em>!
      </span>
    </p>
    <h2 id="movie2">Forbidden Planet</h2>
    <p>Playing at 5:00pm, 9:00pm.</p>
  </body>
</html>
```



Sharpen your pencil Solution

以下是一个歌曲播放列表的HTML，只不过这个列表为空。你的任务是完成下面的JavaScript，向这个列表增加歌曲。下面给出我们的答案。



如果JavaScript能正常运行，加载页面后会看到类似这样的结果。

```
<!doctype html>
<html lang="en">
<head>
  <title>My Playlist</title>
  <meta charset="utf-8">
  <script>
  function addSongs() {
    var song1 = document.getElementById("song1");
    var song2 = document.getElementById("song2");
    var song3 = document.getElementById("song3");

    song1.innerHTML = "Blue Suede Strings, by Elvis Pagely";
    song2.innerHTML = "Great Objects on Fire, by Jerry JSON Lewis";
    song3.innerHTML = "I Code the Line, by Johnny JavaScript";
  }
  window.onload = addSongs;
  </script>
</head>
<body>
  <h1>My awesome playlist</h1>
  <ul id="playlist">
    <li id="song1"></li>
    <li id="song2"></li>
    <li id="song3"></li>
  </ul>
</body>
</html>
```

这些代码可以让播放列表正常工作。

完全可以换成你喜欢的其他歌曲！

上面的代码从DOM获取各个元素，并把innerHTML设置为歌曲名，来设置这些<li>元素的内容。

Sharpen your pencil Solution

下面的Web页面中包含一个列表，其中元素为空，等着你用JavaScript填入温度。我们已经给出了大部分代码，你的任务是完成这个代码，使它将各个列表项的内容设置为数组中相应的温度。你得到加分了吗？下面是我们的答案。

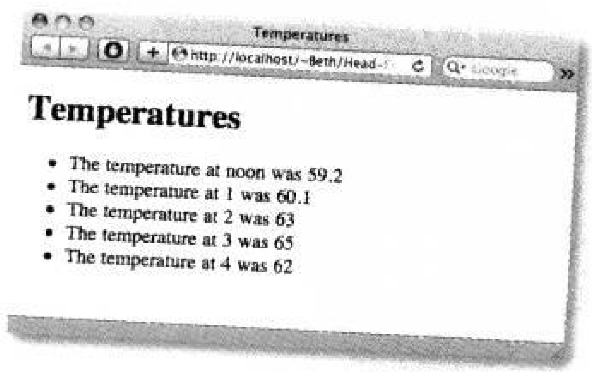
```
<!doctype html>
<html lang="en">
<head>
<title>Temperatures</title>
<meta charset="utf-8">
<script>
function showTemps() {
    var tempByHour = new Array();
    tempByHour[0] = 59.2;
    tempByHour[1] = 60.1;
    tempByHour[2] = 63;
    tempByHour[3] = 65;
    tempByHour[4] = 62;
    for (var i = 0; i < tempByHour.length; i++) {
        var theTemp = tempByHour[i];
        var id = "temp" + i;
        var li = document.getElementById(id);
        if (i == 0) {
            li.innerHTML = "The temperature at noon was " + theTemp;
        } else {
            li.innerHTML = "The temperature at " + i + " was " + theTemp;
        }
    }
}
window.onload = showTemps;
</script>
</head>
<body>
<h1>Temperatures</h1>
<ul>
    <li id="temp0"></li>
    <li id="temp1"></li>
    <li id="temp2"></li>
    <li id="temp3"></li>
    <li id="temp4"></li>
</ul>
</body>
</html>
```

在这里，我们创建了一个新Array存放温度。

在这里结合循环和数组。注意我们使用i作为数组索引，所以每次循环i递增时就能分别访问数组的各个元素。

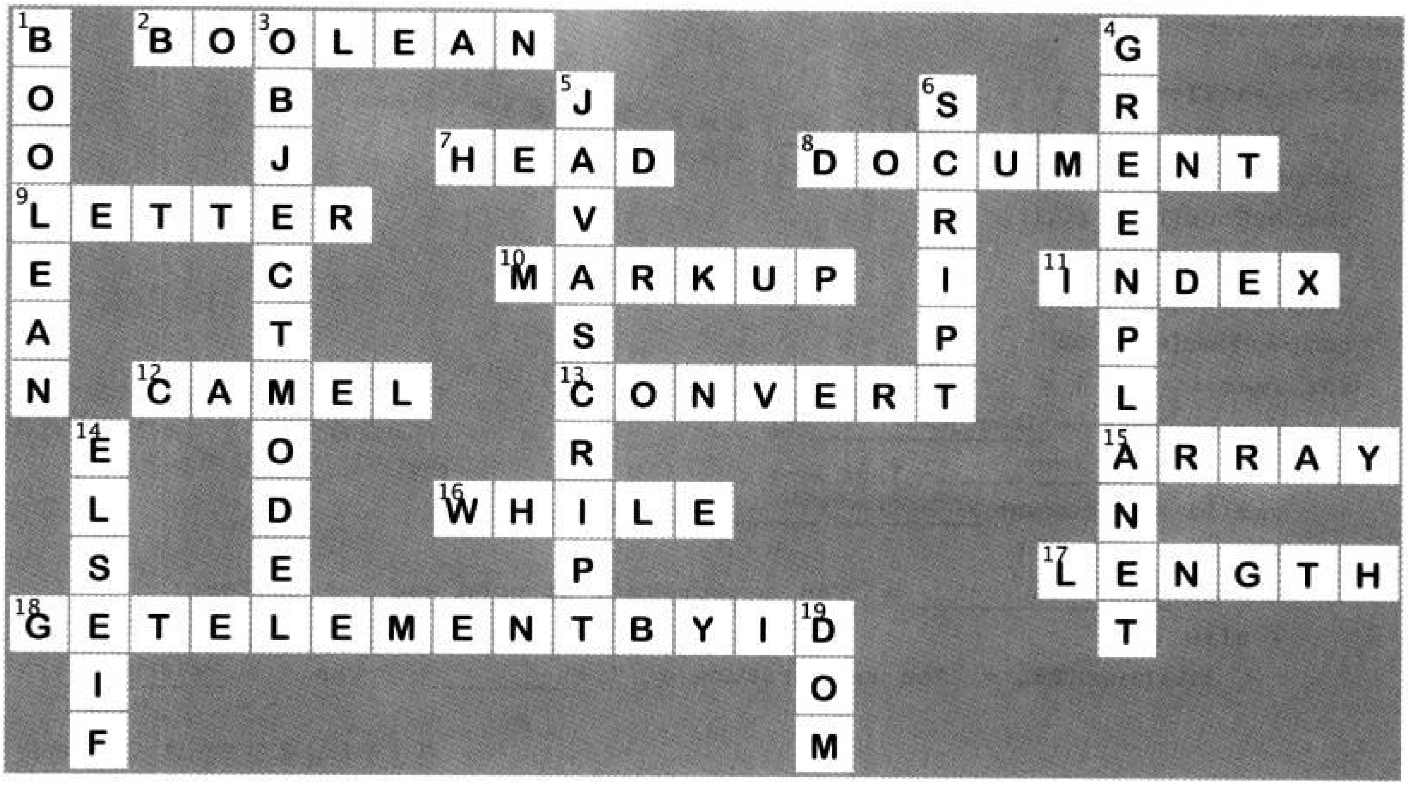
在这里创建要使用的串，这里用到了变量i和theTemp。

这是我们的结果！



pin51.com 拼吾爱

HTML5填字游戏答案



pin51.com 拼吾爱 最新最全资源下载站

3 事件、处理程序，诸如此类

一点点交互

嗯，他看起来不错，不过，如果他不时有些真正的举动，我们的关系会不会更有发展？



要个男人还是人体模特？由你来决定。

你还没有与用户接触呢。你已经了解了JavaScript的基础知识，不过你能与你的用户交互吗？如果页面能响应用户的输入，它们就不再只是文档了，而是有生命的、有反应的应用。在这一章中，你将学习如何处理一种用户输入形式[译注]（嘿嘿，这是个双关语），并把老式的HTML <form>元素关联到具体的代码。听上去有些危险，不过这样确实功能强大。系上安全带，这一章速度会很快，我们将从零出发，转瞬间就会到达交互式应用。

译注：form不仅有“形式”的含义，在这里还隐含了“表单”。

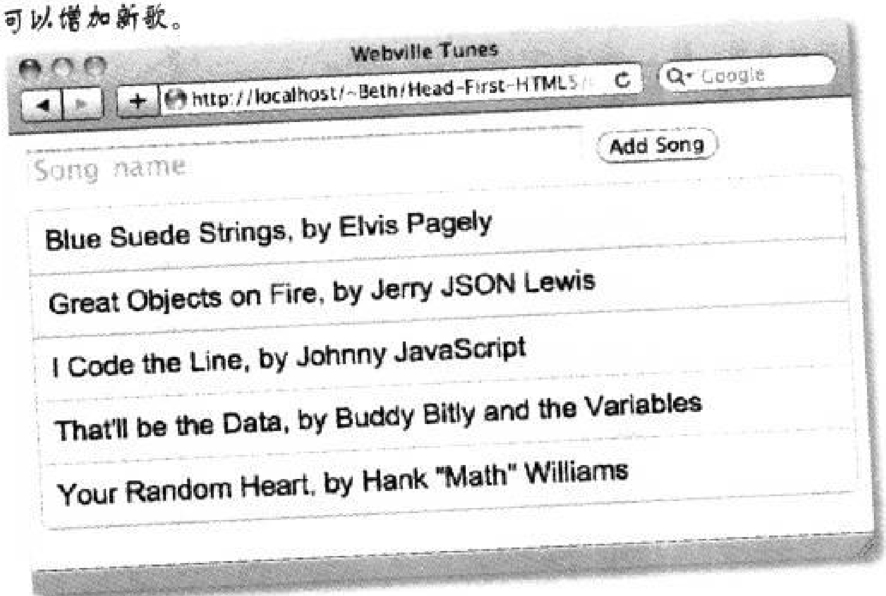
准备进入Web镇之声



pin51.com

嗯，到目前为止，这本书已经让你了解了JavaScript的很多基础知识，尽管我们一直在大谈构建Web应用，但确实还没有具体展示怎么做。所以，下面来点正式的（呵呵，也不一定！我们总在这么说）！构建一个真正的Web应用。

来个播放列表管理器怎么样？给它起个有创意的名字，比方说……嗯，Web镇之声（Webville Tunes）。



BRAIN POWER

假设你已经知道这个代码要做什么：

```
window.onload = init;
```

能不能猜出这个代码可能做什么？

```
button.onclick = handleButtonClick;
```

出发……

出发前我们不需要创建一个庞大、复杂的Web页面。实际上，起步可以很简单。只需要创建一个HTML5文档，其中包含一个表单，还有一个列表（list）元素包含播放列表：

```
<!doctype html>
<html lang="en">
<head>
  <title>Webville Tunes</title>
  <meta charset="utf-8">
  <script src="playlist.js"></script>
  <link rel="stylesheet" href="playlist.css">
</head>
<body>
  <form>
    <input type="text" id="songTextInput" size="40" placeholder="Song name">
    <input type="button" id="addButton" value="Add Song">
  </form>
  <ul id="playlist">

  </ul>
</body>
</html>
```

就是标准的HTML5首部和体。

要把所有JavaScript放在playlist.js文件中。

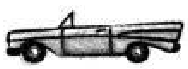
包含了一个样式表，让这个播放列表应用有一个漂亮的外观。*

只需要一个简单的表单。这里有一个文本域，用来输入你的歌曲。我们使用了HTML5的placeholder属性，它会显示一个例子，指示这个输入域中要输入什么。

这里有一个按钮，id为“addButton”，用来向播放列表提交你增加的新歌。

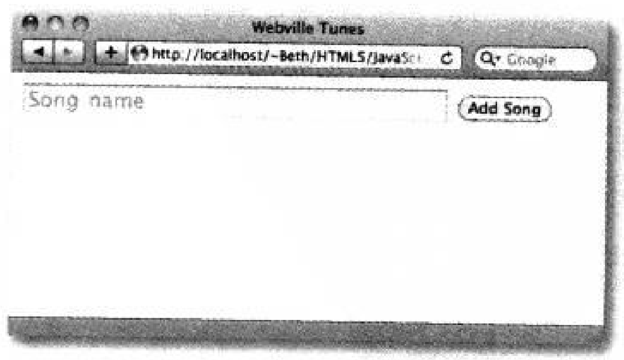
我们将使用一个列表存放这些歌曲。现在这个列表为空，不过，稍后就会用JavaScript代码修改……

试一试



输入上面的代码，把它加载到你最喜欢的浏览器，翻到下一页之前先试着运行一下。

你应该会看到这个结果。



*记住，可以从http://wickedlysmart.com/hfhtml5下载这个样式表（和所有代码）。

pin51.com 拼吾爱

不过我点击“Add Song”按钮时什么也没有发生

嗯，也对也不对。看上去什么也没有发生，但实际上你的浏览器已经知道你点击了这个按钮（取决于你的浏览器，有些浏览器上可能会看到按钮已经按下）。

真正的问题是如何在点击按钮时让它具体做点什么？这个问题的实际含义是，点击按钮时如何调用一些JavaScript代码？

需要两个方面：
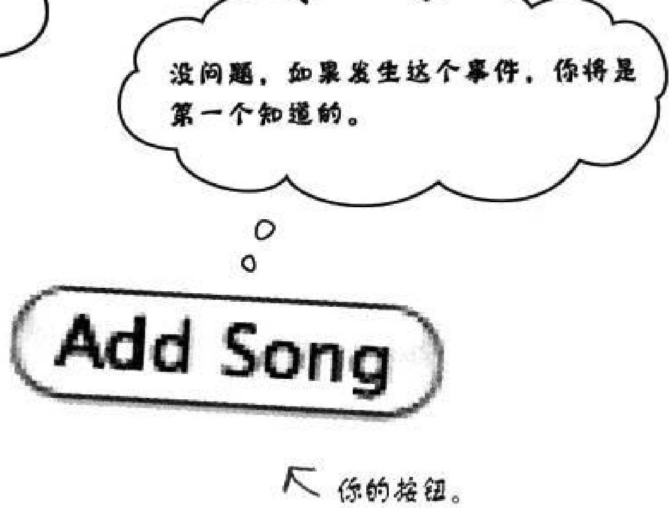
1. 需要一些JavaScript代码，可以在用户点击“Add Song”按钮时执行。这个代码将向播放列表增加一首歌（我们写过这个代码）。

2. 需要一种方法关联这些代码，从而当点击按钮时JavaScript知道要运行你的“增加歌曲”代码。



“用户点击一个按钮时（或者在基于手势的设备上触摸一个按钮时），我们想了解这个按钮。我们感兴趣的是“按钮就是点击事件”。”





处理事件

你会看到，显示页面时，浏览器中会发生很多事情，会点击按钮、代码从网络请求的额外数据会到达，定时器可能到期（后面会介绍这个内容）。所有这些都会导致事件发生，也就是按钮点击事件、数据到达事件、时间到期事件等（当然还有很多很多）。

只要有事件发生，你的代码就有机会进行处理。也就是说，要提供一些代码在事件发生时来调用。现在还不需要你处理任何事件，不过如果你希望当这些事件出现时发生一些有意思的事情，就需要着手处理了。比方说，发生按钮点击事件时，你可能想向播放列表增加一首新歌。新数据到达时，你可能想处理这个数据，并在页面上显示。定时器触发时，你可能想告诉用户他们的前排贵宾票快到期了，如此等等。

所以可以知道，我们想要处理这个按钮点击事件，下面来看如何处理。

pin51.com 拼吾爱

制订计划……

深入到处理程序和事件之前，我们暂且后退一步。这里的目标是点击“Add Song”按钮，向页面上的一个播放列表增加一首歌。这个任务可以这样解决：

1. 建立一个处理程序，用来处理用户点击“Add Song”按钮。
2. 编写处理程序，得到用户输入的歌曲名，然后……
3. 创建一个新元素包含这个新歌，接下来……
4. 把这个元素增加到页面的DOM。

如果你还不理解这些步骤，也不用担心，我们会逐步向你解释…… 对现在来说，只需要对这些步骤有点认识，并按这些步骤编写处理程序就可以了。打开一个新文件（比如playlist.js）来存放你的所有JavaScript代码。

访问“Add Song”按钮

要想让按钮通知我们出现了一个点击事件，首先需要访问这个按钮。很幸运，按钮是用HTML标记创建的，这说明……你应该能够猜到，它会在DOM中用元素表示，而且你已经知道如何得到DOM中的元素。如果返回去查看HTML，可以看到我们将这个按钮的id指定为addButton。所以可以使用getElementById得到这个按钮的引用：

```
var button = document.getElementById("addButton");
```

现在只需要为这个按钮提供一些代码，在出现点击事件时调用。为此，我们要创建一个函数，名为handleButtonClick，它会处理这个事件。稍后就会深入讨论函数。对现在来说，先给出这个函数：

这个函数名为 handleButtonClick，后面会具体介绍这个语法。

函数提供了一个途径可以把代码打包到一个块中。你可以指定一个函数名，并在需要的任何时间重用这个代码块。

```
function handleButtonClick() {
    alert("Button was clicked!");
}
```

目前，调用这个函数时我们要显示一个提醒。

将调用函数时希望执行的所有代码放在大括号中。

1. 建立一个处理器处理用户的点击事件
2. 编写处理程序，得到歌曲名
3. 创建一个新元素包含这首新歌
4. 将这个元素增加到页面的DOM

为按钮指定一个点击处理程序

好了，我们已经有了一个按钮，也有了作为处理程序的函数handleButton-Click，下面把它们关联在一起。为了做到这一点，要使用按钮的一个属性onclick。如下设置onclick属性：

```
var button = document.getElementById("addButton");
button.onclick = handleButtonClick;
```

调用getElementById之后得到了按钮，接下来把onclick属性设置为出现点击事件时我们想要调用的函数。

你可能还记得，我们曾使用window.onload属性在窗口加载之后调用一个函数，那时的做法与现在类似。不过，这里是在点击按钮时调用函数。现在把它们关联在一起：

```
window.onload = init;
function init() {
    var button = document.getElementById("addButton");
    button.onclick = handleButtonClick;
}
function handleButtonClick() {
    alert("Button was clicked!");
}
```

类似于上一章的做法，我们将使用一个init函数，等页面完全加载时才会调用并执行这个函数。

页面加载之后，我们将获取按钮，并设置它的onclick处理程序。

这个点击处理程序会在我们点击按钮时显示一个提醒。

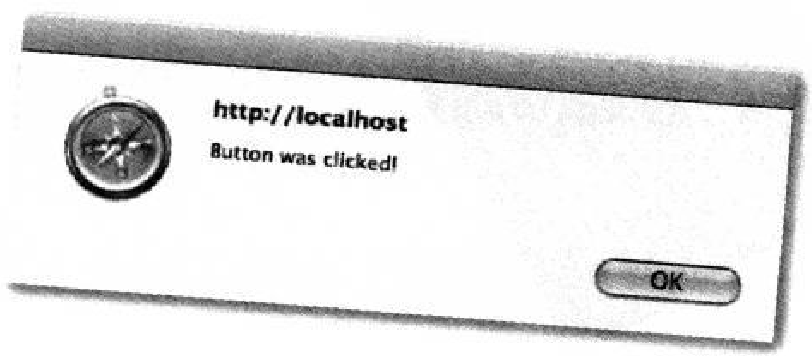
试一试……



输入上面的代码（放在playlist.js文件中），加载页面，随便多次点击这个按钮，每次点击你都会看到一个提醒。

测试这个新的按钮点击处理程序之后，可以坐下来，研究一下代码，想一想它到底是怎么做的。

如果你认为已经想清楚了，可以翻开下一页，我们会逐步详细介绍，确保你真正记住。



pin51.com 拼吾爱 最新最全资源分享下载站

仔细研究发生了什么……
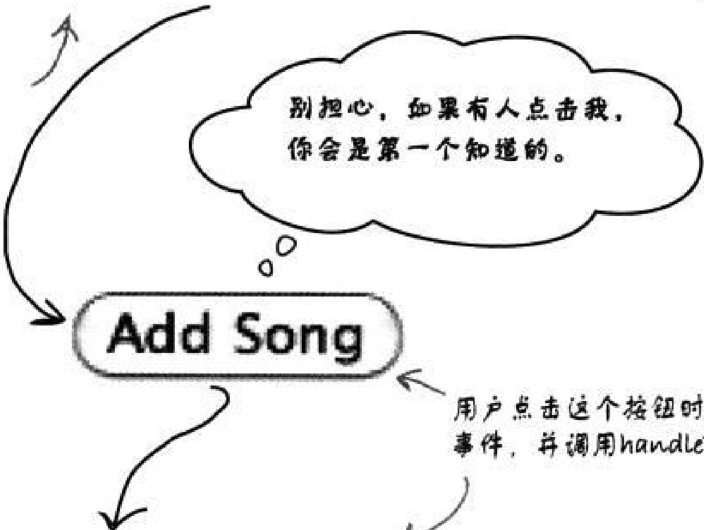
前面几页介绍了很多新想法，下面再逐步分析这个代码，确保牢牢地把它记住。开始吧：

1 你做的第一件事是在HTML表单中放入一个按钮。有了按钮，还需要一种方法可以捕获用户点击这个按钮，从而能执行一些代码。为此，我们创建了一个处理程序，并把它赋至按钮的onclick属性。

```
function init() {
    var button = document.getElementById("addButton");
    button.onclick = handleButtonClick;
}
```

在init函数中设置了按钮点击事件处理程序（也就是说，在页面完成加载之后）。

button对象有一个onclick属性，我们把这个属性设置为handleButtonClick函数。



用户点击这个按钮时，会触发“点击”事件，并调用handleButtonClick函数。

```
function handleButtonClick() {
    alert("Button was clicked!");
}
```

2 你还写了一个简单的处理程序，告诉用户这个按钮被点击。稍后我们会为这个处理程序编写真正的代码，不过目前这个代码很适合测试。



包含代码的处理程序

3 已经编写了代码，页面已经由浏览器加载并显示，并且安装了处理程序……现在万事俱备，只欠东风，全看用户的了……

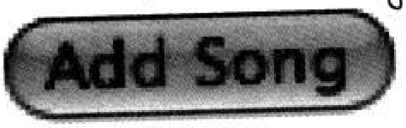
哈……点击按钮……没问题……



4 最后，用户点击这个按钮，按钮开始起作用，注意到它有一个处理程序，并调用这个处理程序……

提醒，用户点击按钮了。

我发现我有一个相应的处理程序，最好让他知道。

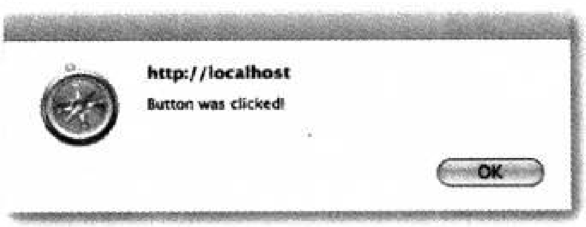


太棒了！有人点击按钮了。我要运行handleButtonClick函数。



```
function handleButtonClick() {
    alert("Button was clicked!");
}
```

他们要我提醒你有人点击了按钮……我知道，用一个提醒对话框有些大材小用，不过，不管怎样，这就是我的本职工作。



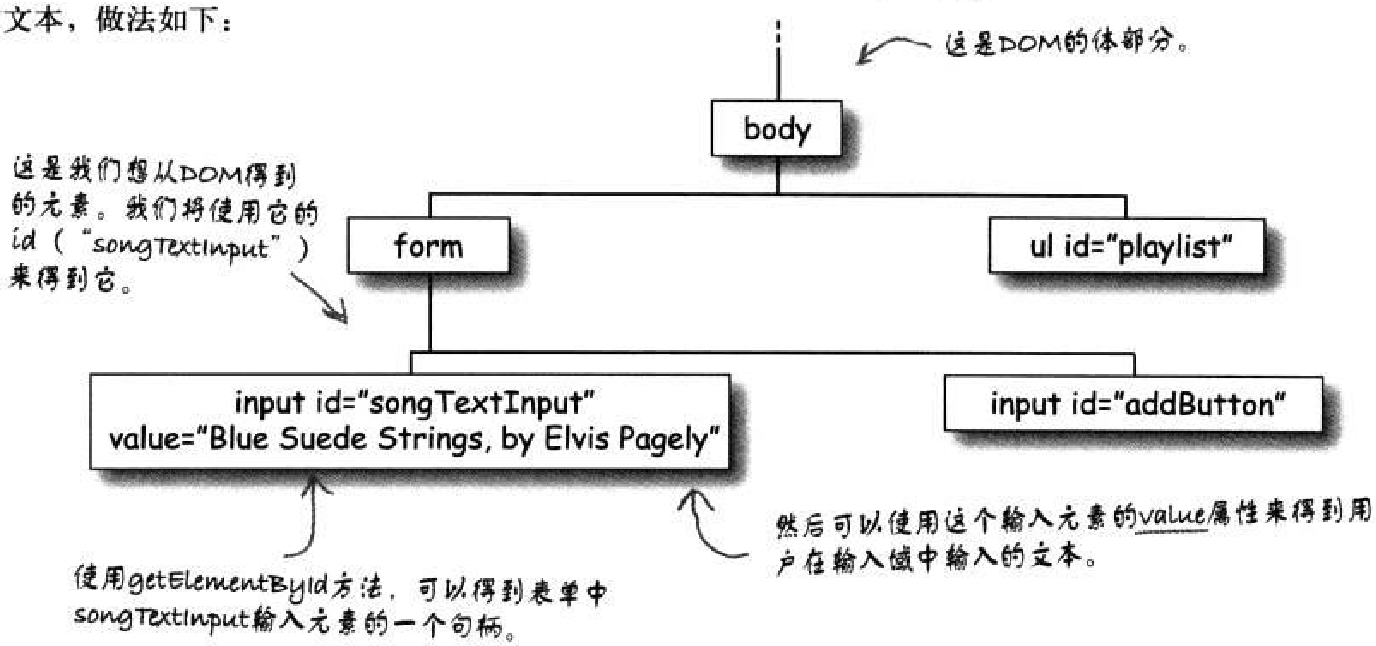
获得歌曲名

1. 建立一个处理器处理用户的点击事件
2. **编写处理程序，得到歌曲名**
3. 创建一个新元素包含这首新歌
4. 将这个元素增加到页面的DOM

我们已经准备进入任务的第二步：获得用户输入的歌曲名。一旦有了歌曲名，就可以考虑如何在浏览器中显示播放列表了。

不过，到底怎么得到歌曲名呢？这是用户输入的东西，对吧？哈，不过Web页面中发生的任何变化都会在DOM中得以反映，所以用户输入的文本肯定也在DOM中。

要从一个表单文本输入元素得到文本，首先必须从DOM得到这个输入元素，你肯定知道该怎么做：没错，要用`getElementById`。另外，一旦得到这个输入元素，可以使用文本输入元素的`value`属性来访问用户在表单域中输入的文本，做法如下：



Sharpen your pencil

修改下面的handleButtonClick函数，得到用户在表单输入元素中输入的歌曲名。对照96页的答案看看你做得对不对。

```
function handleButtonClick() {
    var textInput = document.getElementById("..............................");
    var songName = .............................. .value;
    alert("Adding " + .............................. );
}
```

pin51.com

Sharpen your pencil 附加题

如果你希望在点击按钮前先确保用户确实输入了一些文本，又该怎么做呢（同样地，可以检查96页上的答案）？

....

....

there are no Dumb Questions

问： **如果用户没有输入任何内容，文本输入域的vaule属性值是什么呢？这个值是null吗？或者如果用户没有输入文本是不是“Add Song”按钮根本不会调用处理程序？**

答： “Add Song”按钮可没那么聪明。如果你想确定用户是不是输入了内容，要由你的代码来处理。另外，要知道文本输入域是否为空（也就是说，用户没有输入任何内容），可以查看它的值是否等于一个不包含任何内容的串，也称为空串，写作“”，就是双引号，而且中间没有任何字符。我们非常理解你为什么会认为它是null，因为我们说过null是一个不包含值的变量的值，不过，在文本输入域看来，它并不是没有包含内容，实际上它包含一个串，只不过这个串中没有内容而已。搞清楚了吧。☺

问： **我以为文本输入域的“value”是一个属性（attribute），但你把它叫做属性（property），为什么？**

答： 你说得对，value确实是HTML文本输入元素的一个属性（attribute）。可以使用value属性初始化一个文本输入元素的值。不过，在JavaScript中，要访问一个用户输入的值，需要使用我们由DOM得到的输入元素的value属性（property）。

问： **除了按钮点击事件外，JavaScript中还能处理哪些事件？**

答： 还有一大堆其他鼠标事件可以处理。例如，可以检测和处理鼠标键按下、鼠标移到一个元素上或者移出一个元素、鼠标拖动，甚至鼠标键按下并保持（与鼠标点击不同）。另外我们还提到过很多其他类型的事件，如新数据到来的事件、定时器事件、与浏览器窗口有关的事件等。在本书后面你还会看到很多其他类型的事件处理。一旦知道如何处理一个事件，所有事件就都能处理了！

问： **等待事件时JavaScript在做什么？**

答： 除非你编写了JavaScript代码来做一些工作，否则它会一直空闲，直至事件发生（用户与界面交互、数据从Web到达、定时器到期等）。这是一件好事。这说明计算机可以处理其他事情，比如让你的浏览器能够响应。本书后面，你还会了解如何创建在后台运行的任务，使你的浏览器在运行任务代码的同时还能对事件做出响应。

pin51.com 拼吾爱

Sharpen your pencil Solution

修改下面的handleButtonClick函数，得到用户在表单输入元素中键入的歌曲名。以下是我们的答案：



首先需要得到表单中文本输入元素的一个引用。我们已经为这个元素指定id为"songTextInput"，所以可以使用这个id由getElementById得到一个引用。

```
function handleButtonClick() {
    var textInput = document.getElementById("songTextInput");
    var songName = textInput.value;
    alert("Adding " + songName);
}
```

文本输入元素的value属性包含文本输入域中输入的内容，这就是一个串。在这里我们把这个文本赋至变量songName。

现在只需弹出一个提醒，这会显示"Adding"和歌曲名。

Sharpen your pencil Solution 附加题

如果你希望在点击按钮前先确保用户确实输入了一些文本，又该怎么做呢？我们的答案如下：

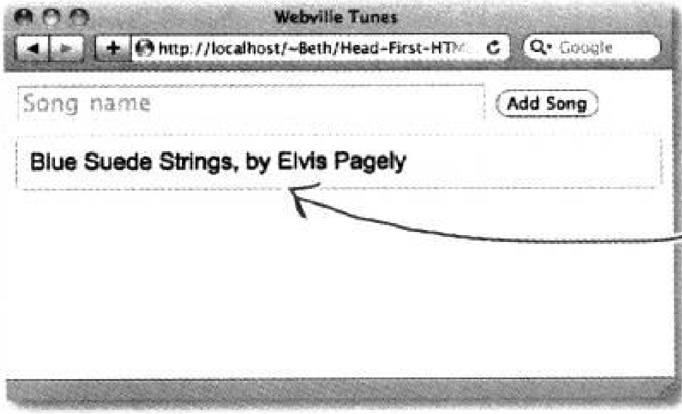
```
function handleButtonClick() {
    var textInput = document.getElementById("songTextInput");
    var songName = textInput.value;
    if (songName == "") {
        alert("Please enter a song");
    } else {
        alert("Adding " + songName);
    }
}
```

可以使用一个if语句，将songName串与一个空串比较，确保用户确实输入了一些文本。如果他们没有输入任何内容，就做出提醒并要求他们输入一首歌。

www.win51.com 最新最全的下载资源分享

如何向页面增加一首歌？

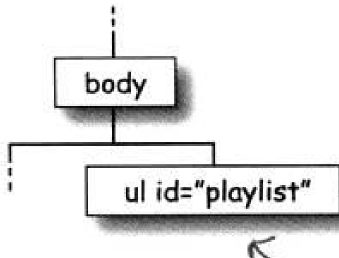
我们已经做了不少事情了！你可以向表单输入一个歌名，点击Add Song按钮，得到你在表单中输入的文本，所有这些都可以用代码完成。现在我们打算把播放列表就显示在页面上。看上去应该是这样的：



点击"Add Song"按钮时，你的JavaScript代码会把这首歌增加到页面上的歌曲列表。

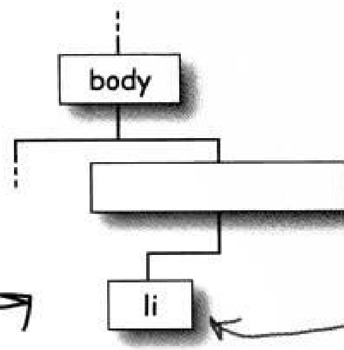
我们要这样做：

① 你可能已经注意到，最早输入HTML时我们已经在HTML标记中放了一个空列表（准确地讲，是一个空的<ul>元素）。因此，目前的DOM会是这样：



这是DOM中的列表，目前它还是空的。

② 每次输入一个新歌时，我们希望向这个无序列表增加一个新列表项。为此，要创建一个新的<li>元素，其中包含歌名。然后将这个新的<li>元素增加到DOM中的<ul>。一旦增加，浏览器会完成它的工作，我们将看到页面更新，就好像<li>一直在那里一样。当然，这些都是用代码完成的。再查看DOM，确保你确实已经了解我们需要做什么。



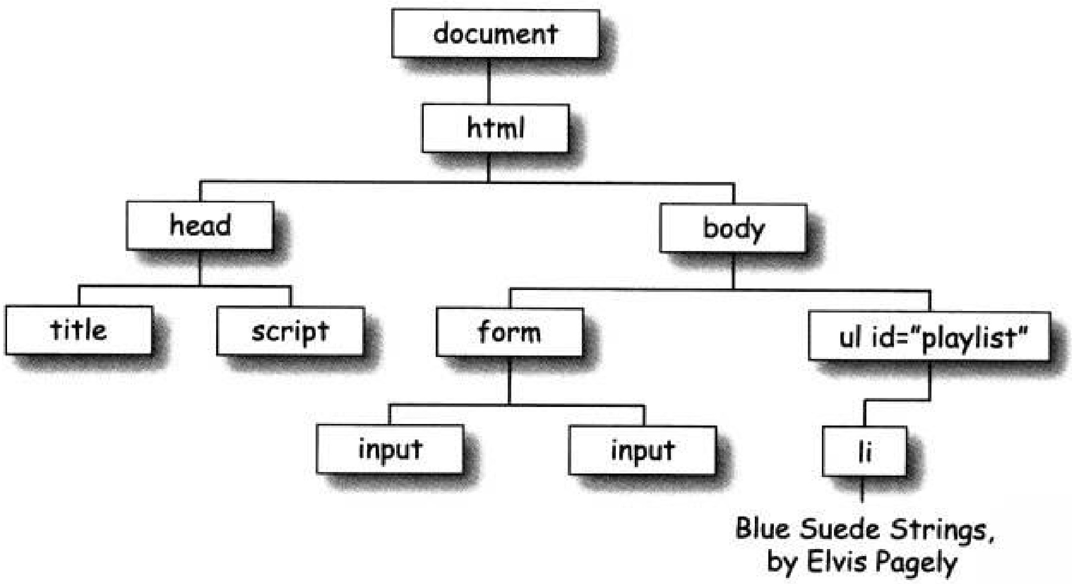
输入一首歌时，要创建一个新的列表项(<li>元素)，并把它增加到<ul>列表。

pin51.com

Exercise



对于这里显示的播放列表，画出增加所有这些歌之后的DOM。注意歌曲增加到页面的顺序，确保DOM中的元素顺序也是正确的。我们已经帮你完成了一个。学习后面的内容之前，请对照检查这一章最后的答案。



在这里为以上播放列表画出DOM的其余部分。

要对<li>元素增加到父元素的顺序做些假设吗？

如何创建一个新元素

你已经看到如何通过DOM访问现有的元素。除此以外，还可以使用DOM创建新元素（下一步还能把这些新元素增加到DOM，这个内容稍后介绍）。



假设我们想创建一个`<li>`元素。做法如下：

使用document.createElement来创建新元素。它会返回新元素的一个引用。

```
var li = document.createElement("li");
```

这里将这个新元素赋至变量li。

把希望创建的元素类型作为一个串传入createElement。

li

createElement会创建一个全新的元素。注意它还会没有插入到DOM。现在它只是一个到处漂游的元素，需要在DOM找到一个位置。

现在有了一个新的`<li>`元素，其中还没有任何内容。你已经知道在元素中加入文本的一种方法：

```
li.innerHTML = songName;
```

我们的li变量。

这会把<li>的内容设置为歌曲名。

li

Blue Suede Strings, by Elvis Pagely

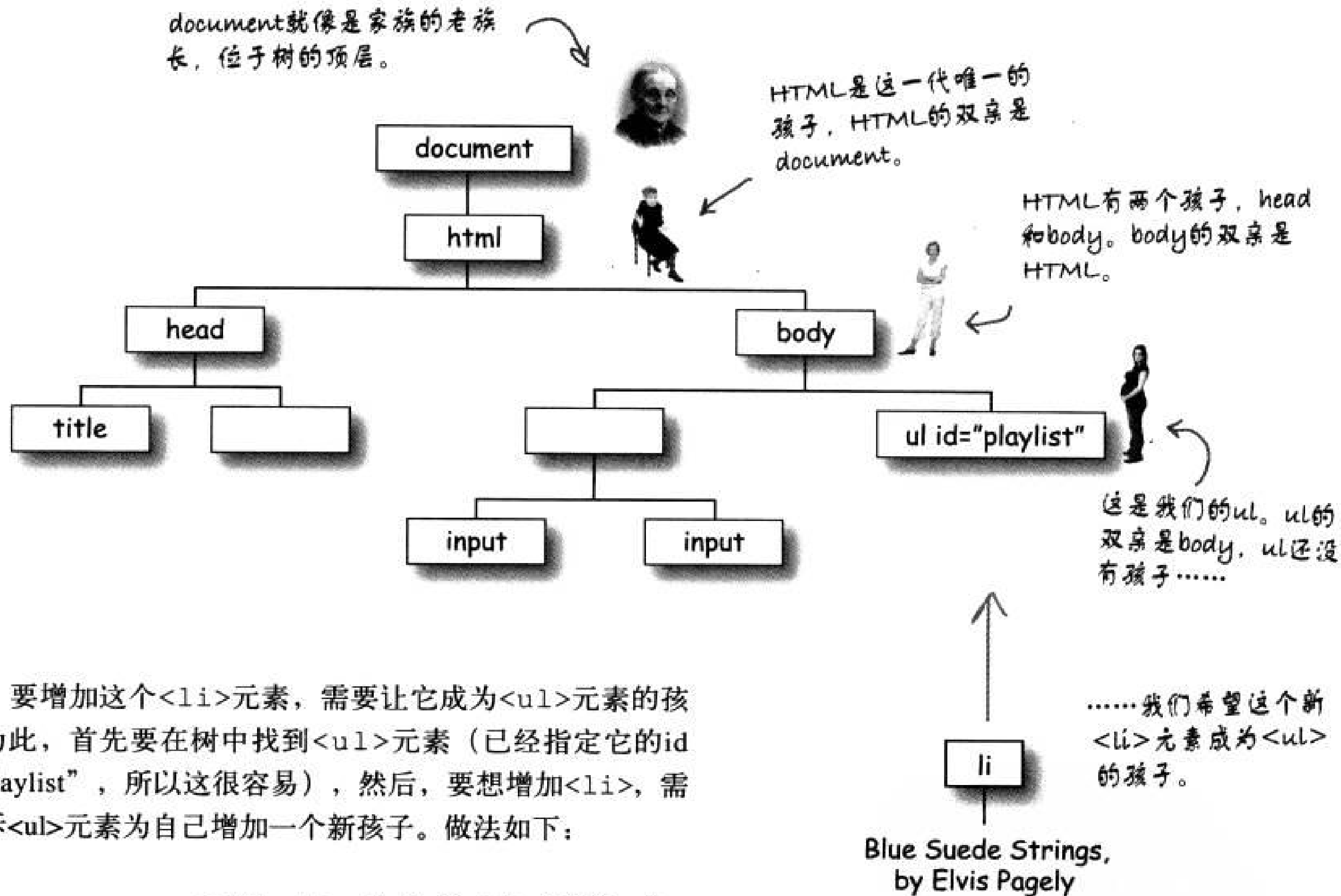
这是我们得到的新的li元素对象。不过它还不是DOM的一部分！

1. 建立一个处理器处理用户的点击事件
2. 编写处理程序，得到歌曲名
3. 创建一个新元素包含这首新歌
4. 将这个元素增加到页面的DOM

www.ln51.com 最新最全编程资源分享下载

向DOM增加新元素

要向DOM增加新元素，必须知道你想把它放在哪里。嗯，我们当然知道要把它放在哪里：这个<li>元素要放在<ul>元素中。不过怎么做呢？再来看看DOM。还记得吧？我们说过它就像一棵树。可以考虑家族树：



所以，要增加这个<li>元素，需要让它成为<ul>元素的孩子。为此，首先要在树中找到<ul>元素（已经指定它的id为“playlist”，所以这很容易），然后，要想增加<li>，需要告诉<ul>元素为自己增加一个新孩子。做法如下：

使用getElementById得到id=“playlist”的<ul>元素的一个引用。

```
var ul = document.getElementById("playlist");
ul.appendChild(li);
```

让<ul>元素增加<li>元素作为它的一个子元素。一旦完成，DOM中<li>就会成为<ul>的一个子节点，浏览器会相应更新显示，反映这个新<li>。

每次调用appendChild时，新的<li>元素增加到<ul>元素时会增加到现有的其他<li>元素的后面。

拼吾爱 最新最全资源分享下载站 pin5i.com

集成在一起……

下面把所有这些代码集成在一起，放在handleButtonClick函数中。如果还没有输入，现在就来输入这些代码以便测试。

```
function handleButtonClick() {

    var textInput = document.getElementById("songTextInput");

    var songName = textInput.value;

    var li = document.createElement("li");

    li.innerHTML = songName;

    var ul = document.getElementById("playlist");

    ul.appendChild(li);

}
```

首先，创建新的<li>元素，歌名将放在这里。

然后，把这个元素的内容设置为歌名。

id为"playlist"的<ul>是新<li>的父元素。所以下面来得到这个元素。

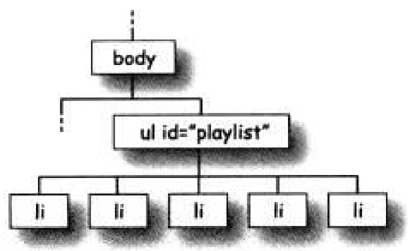
然后使用appendChild把li对象增加到ul。

注意，我们要求父元素ul增加li作为它的一个新孩子。

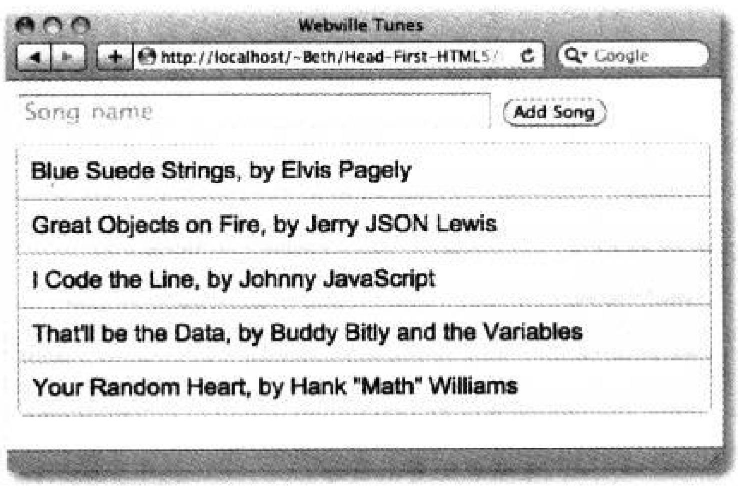
……试一试



让"Web镇之声"派上用场，增加一些歌曲。以下是我们得到的结果。



增加了所有这些新<li>元素之后，DOM如上。



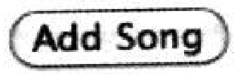
现在，我们输入一个歌名并点击按钮时，这首歌就会增加到DOM，所以我们会看到页面改变，这首新歌会出现在列表中。

pin51.com 拼吾爱

回顾我们做的工作

这一章你做了很多事情（尽管时间很短！）。你建立了一个播放列表应用，可以用来输入一首歌、点击一个按钮，然后把这首歌增加到页面上的一个列表，所有这些都使用JavaScript代码完成。

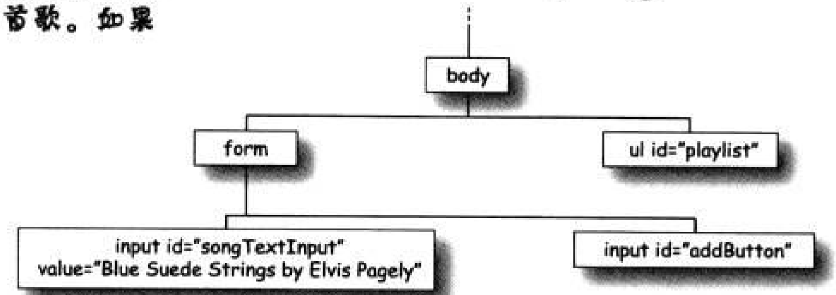
1. 首先你建立了一个事件处理程序，用来处理用户点击“Add Song”按钮的事件。你创建了一个函数handleButtonClick，并把“Add Song”按钮的onclick属性设置为这个函数。





用户点击“Add Song”按钮时，会调用handleButtonClick处理程序。

2. 接下来，为按钮点击事件处理程序编写代码，从输入文本域得到歌名。你使用input.value属性来得到文本，甚至还增加了一个检查，确保用户确实输入了一首歌。如果用户没有输入，你就提醒他们。



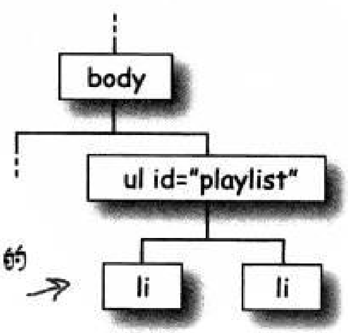
在handleButtonClick中，你会得到用户键入的歌名，通过使用input.value属性从DOM得到文本。

3. 要把这首歌增加到播放列表，你使用document.createElement创建了一个<li>元素，并使用innerHTML把这个元素的内容设置为歌名。



创建一个新的<li>元素，并设置这个<li>元素的内容为歌名。

4. 最后，把这个新<li>元素作为<ul>父元素的子元素增加到DOM。这里使用了appendChild，告诉<ul>元素“追加<li>元素作为一个子元素”，从而把它增加到DOM。元素增加到DOM时，浏览器会更新用户看到的页面，播放列表中将包含这首歌。



在DOM中增加一个新的子元素会更新页面。

等一下，我知道我们确实在和DOM等打交道，
不过这算一个真正的Web应用吗？如果我把浏览器关掉，
所有歌就都不见了。如果这是一个真正的应用，我的播放
列表项是不是该一直保留啊？



同意，播放列表确实应当持久保存。

毕竟，如果歌曲不能保留，输入这些歌又有什么意义？另外，你可能还想增加很多其他功能。例如，你可能想使用音频/视频API增加一个音频接口，使你能真正听歌；可能想使用一个Web服务与朋友分享歌曲（如Facebook和Twitter）；可能还想找到当地其他喜欢相同艺术家的人（使用地理定位API），相信你肯定还有更多想法。

不过先来看这个播放列表……我们只是想让你热热身，构建一个小的交互式应用，这个播放列表应用就很合适。另外，存储歌曲需要用到HTML5 Web存储API，好几章之后才会介绍。

嗯，另一方面，我们确实不想在这里有所隐瞒……

翻到下一页。

pin51.com 最新编程资源分享下载站 拼吾爱

成品代码



我们已经为你做好了一些代码，可以用来保存你的播放列表。对现在而言，你只需要输入这些代码，另外对你原先的代码做两处很小的改动，你就会得到一个HTML5存储的播放列表了。

“Web存储”一章中我们会介绍在浏览器中本地存储信息的所有详细内容，不过，现在先把这个播放列表运转起来。

当然，仔细查看这个成品代码也没有坏处。你可能会惊奇地发现你已经了解不少，另外很多内容即使不了解，你也能猜出个大概。

Watch it!

这个成品代码在IE 6或7中不能用。

IE 6和7不支持localStorage。所以如果你使用IE，一定要用版本8或更高版本。

Watch it!

如果你从file://提供页面，而不是从一个类似localhost://的服务器或联机托管服务器提供，在有些浏览器上这个成品代码可能不能正常工作。

我们会在后面一些章节中处理这种情况（对于新的HTML5特性，这种情况经常出现）。对现在来说，如果你不希望运行一个服务器或者不想把文件复制到一个联机的托管服务器，可以先使用Safari或Chrome。

如何增加成品代码……



成品代码

下面的成品代码可以增加到你的“Web镇之声”应用中，能把你创建的完美播放列表保存起来。你要做的就是建立一个新文件，playlist_store.js，输入下面的代码，然后对你现有的代码做两处改动（见下一页）。

```
function save(item) {
    var playlistArray = getStoreArray("playlist");
    playlistArray.push(item);
    localStorage.setItem("playlist", JSON.stringify(playlistArray));
}

function loadPlaylist() {
    var playlistArray = getSavedSongs();
    var ul = document.getElementById("playlist");
    if (playlistArray != null) {
        for (var i = 0; i < playlistArray.length; i++) {
            var li = document.createElement("li");
            li.innerHTML = playlistArray[i];
            ul.appendChild(li);
        }
    }
}

function getSavedSongs() {
    return getStoreArray("playlist");
}

function getStoreArray(key) {
    var playlistArray = localStorage.getItem(key);
    if (playlistArray == null || playlistArray == "") {
        playlistArray = new Array();
    }
    else {
        playlistArray = JSON.parse(playlistArray);
    }
    return playlistArray;
}
```

将这些代码输入“playlist_store.js”。

pin51.com

集成成品代码



还需要做几个小调整来集成这个存储代码。首先，在在playlist.html的<head>元素中增加一个指向playlist _ store.js的引用。

```
<script src="playlist_store.js"></script>
<script src="playlist.js"></script>
```

把它增加到playlist.js链接上面。这会加载成品代码。

现在只需要向playlist.js增加两行代码，分别加载和保存播放列表：

```
function init() {
    var button = document.getElementById("addButton");
    button.onclick = handleButtonClick;
    loadPlaylist();
}

function handleButtonClick() {
    var textInput = document.getElementById("songTextInput");
    var songName = textInput.value;
    var li = document.createElement("li");
    li.innerHTML = songName;
    var ul = document.getElementById("list");
    ul.appendChild(li);
    save(songName);
}
```

这样一来，当你加载页面时就从localStorage加载已存储的歌曲，使你能看到之前保存的歌曲。

每次向播放列表增加一首歌时，这会把它保存起来。

试一试保存的歌曲



好了，重新加载页面，输入一些歌曲。退出浏览器。再次打开浏览器，并加载这个页面。你会看到存储的所有歌曲都安全地出现在播放列表中。

嗯，你是不是已经厌倦你的播放列表了？想把它删除？可以参考"Web存储"一章！



增加这些歌曲，关闭浏览器，重新打开浏览器，加载页面，哈，它们还在那里。

pin51.com 拼吾爱 最新最全资源分享下载站



太对了，事不宜迟。

我们确实希望带你完成一个完整的交互式例子，结合HTML标记和JavaScript共同构建一个Web应用的第一部分。想想看，实际上你已经做了不少了：

(1) 你已经向页面插入了代码。

(2) 建立了一个按钮点击事件，而且编写了代码来捕获和处理按钮点击事件。

(3) 向DOM请求了信息。

(4) 创建了新元素并且增加到DOM。

还不错！既然你现在对这些内容如何结合有了一些直观的认识，下面再朝JavaScript方向走得更远一点，看看函数和对象究竟是怎么工作的。

不过，这可不是常规的旅行，绝对不是，我们会掀开阴井盖，偷偷瞄一眼Web镇的函数。

有兴趣吗？来吧，和我们一起进入第4章……

pin51.com 拼吾爱 最新最全最好的资源下载站

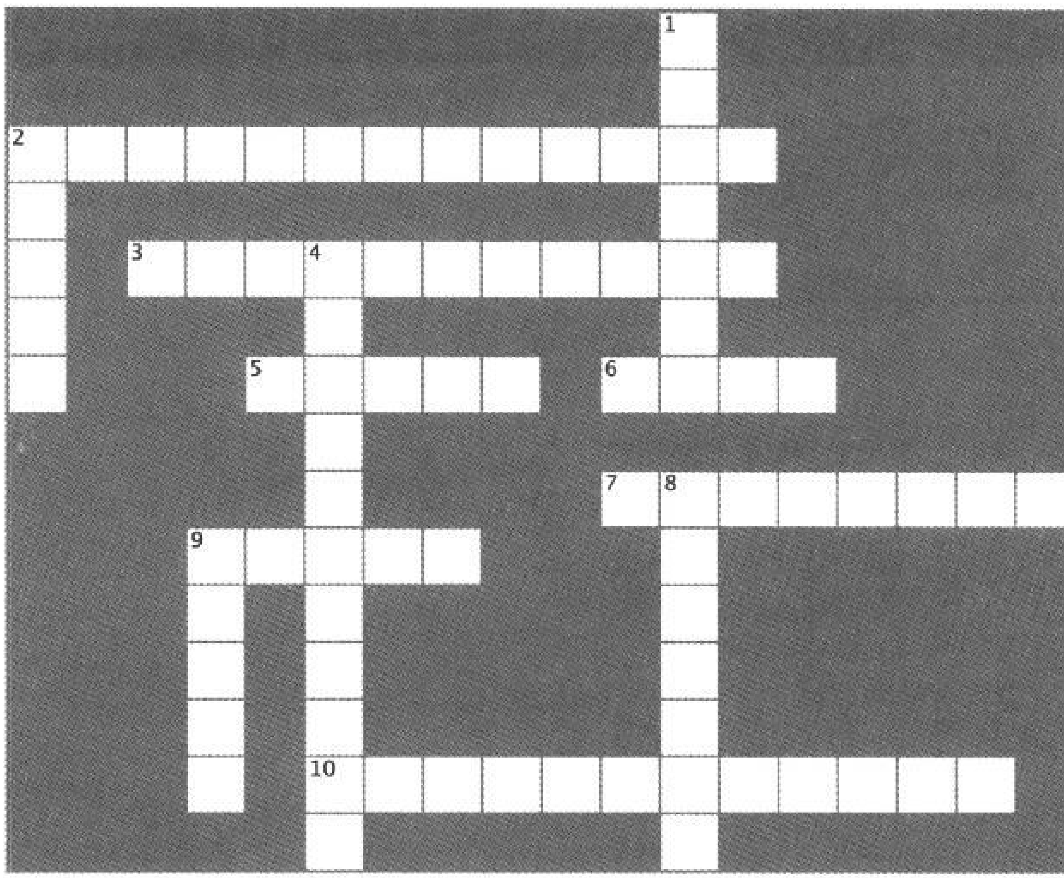
BULLET POINTS

- 你的浏览器中一直在发生很多事件。如果你想对这些事件做出响应，就要用事件处理程序来处理这些事件。
- 点击Web页面上的一个按钮时会触发按钮点击事件。
- 处理一个按钮点击事件时，要注册一个函数来处理这个事件。要做到这一点，需要编写一个函数，并把按钮的onclick属性设置为这个函数名。
- 如果注册了一个按钮点击事件处理程序，点击这个按钮时就会调用这个函数。
- 要编写函数代码作为处理程序对按钮点击事件做出响应。可以提醒用户或者更新页面，也可以做其他响应。
- 要得到用户在一个表单输入文本域中输入的文本，要使用这个输入域的value属性。
- 如果用户没有向表单输入文本域输入任何内容，这个域的值将是空串(“”)。
- 可以用一个if测试和==来比较变量和空串是否相等。
- 要把一个新元素增加到DOM，首先需要创建这个元素，然后增加这个元素作为某个元素的子元素。
- 使用document.createElement可以创建一个新元素。将标记名（例如，“li”）传入函数调用，指示要创建的元素。
- 要在DOM中增加一个元素作为某个父元素的子元素，需要得到这个父元素的引用，并对父元素调用appendChild，传入要增加的子元素。
- 如果使用appendChild向一个父元素增加多个子元素，每个新的子元素会追加到其他子元素的后面，所以它们会出现在页面中其他子元素的后面或下面（假设没有用CSS改变页面布局）。
- 可以使用Web存储API（localStorage）在用户的浏览器中存储数据。
- 我们使用了localStorage保存播放列表中的歌曲，这里采用了成品代码。你会在第9章了解更多有关localStorage的内容。
- 下一章你更多地了解DOM和JavaScript特性，比如函数和对象。



HTML5填字游戏

给自己留点时间来消化学过的内容，好好理解HTML和JavaScript之间的交互。考虑它们如何结合。考虑的同时，可以做做这个填字游戏，复习一下有关内容。所有单词都选自这一章。



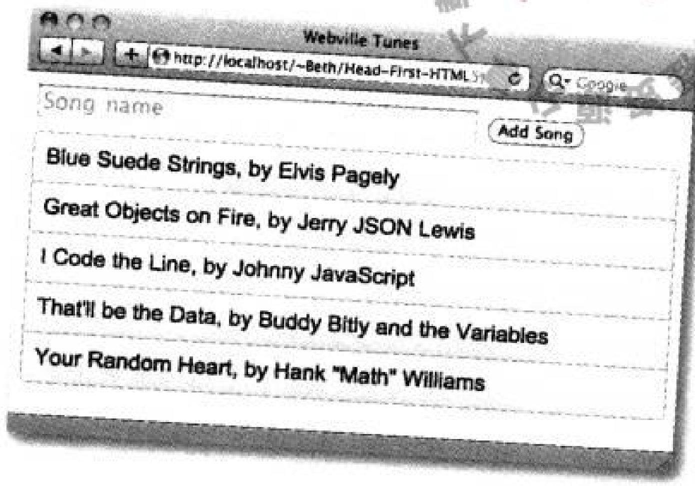
横向

2. DOM用来创建新元素的方法。
3. DOM用来增加新元素的方法。
5. 用户点击按钮时发生的事情。
6. DOM就像一个家族________。
7. DOM树的老族长。
9. 如果用户没有输入任何内容，则表单输入元素的默认值是一个_______串。
10. 成品代码中用来启用存储。

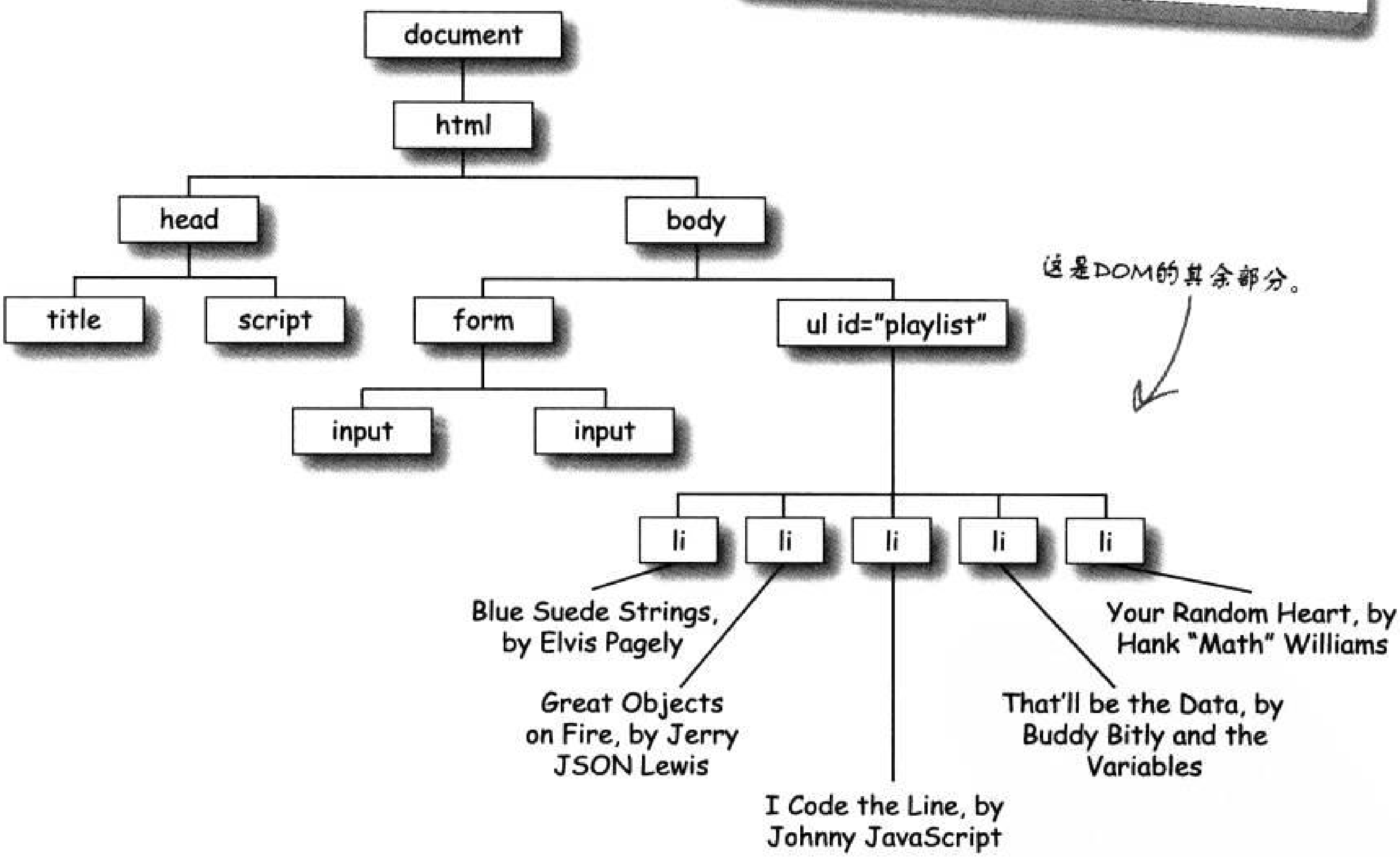
纵向

1. 负责处理事件的代码。
2. 插入新元素作为一个_________。
4. 我们的示例歌曲中用到的艺术家。
8. 接下来要介绍什么？函数和_________。
9. 按钮点击是一个___________。

Exercise Solution



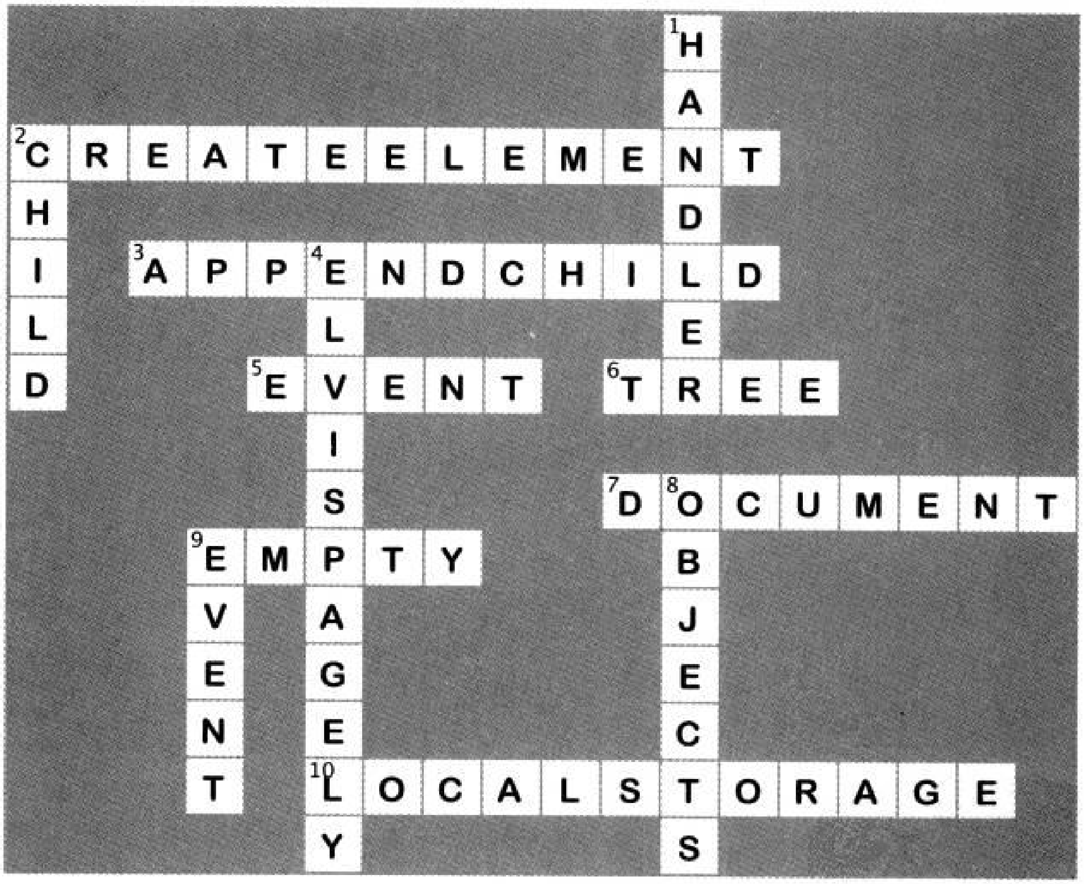
对于这里显示的播放列表，画出增加所有这些歌之后的DOM。注意歌曲增加到页面的顺序，确保DOM中的元素顺序也是正确的。以下是我们的答案。



要对<li>元素增加到父元素的顺序做些假设吗？

是的，因为这会影响歌曲在页面上的显示顺序。appendChild总是把新元素追加到现有子元素的后面。

HTML5填字游戏答案



pin5i.com
拼吾爱
最新编程资源分享下载站

pin51.com 拼吾爱 最新最全电子书免费下载站

4 JavaScript函数和对象

正式JavaScript



你能把自己称做脚本开发人员吗？可能吧，你已经懂得不少JavaScript了。不过，既然能作为一个程序员，谁还只想当个区区脚本开发人员呢？现在严肃一点，是时候了,你该学习学习**函数**和**对象**。要想编写更强大、更有组织，而且更易于维护的代码，它们正是关键所在。另外，HTML5 JavaScript API中也大量使用了函数和对象，所以你对它们越了解，就能越快地熟悉新API并熟练掌控。系上安全带，这一章要求你必须全力以赴……

扩展你的词汇



你已经可以用JavaScript做过很多事情，下面来看看你了解的一些内容：

```
<script>
  var guessInput = document.getElementById("guess");
  var guess = guessInput.value;
  var answer = null;

  var answers = [ "red",
                  "green",
                  "blue"];

  var index = Math.floor(Math.random() * answers.length);

  if (guess == answers[index]) {
      answer = "You're right! I was thinking of " + answers[index];
  } else {
      answer = "Sorry, I was thinking of " + answers[index];
  }
  alert(answer);
</script>
```

从文档对象模型获取一个元素。

得到一个表单输入文本域的值。

创建一个新数组，并填入字符串。

使用函数库。

得到数组的一个属性，比如length。

根据条件做出判断。

使用数组的元素。

使用浏览器函数，比如alert。

不过，到目前为止，你了解的很多知识都是不正式的。当然，你可以从DOM得到一个元素，还能为它指定一些新HTML，但是如果让你从理论上准确地解释document.getElementById做了什么，嗯，这可能有点难度。不用担心，等你读完这一章，就能轻而易举地回答这种问题了。

为了达到这个目的，我们并不打算一上来就对getElementById做一个深入的理论分析。这可不是我们的想法，我们希望更有意思一点：先来扩展JavaScript的词汇，用来做一些新的工作。

www.minxue.com 拼吾爱 最新最全资源分享 下载站

如何增加你自己的函数

你一直在用内置函数，比如alert，甚至Math.random，不过如果想增加你自己的函数呢？假设我们想写类似下面的代码：

```
var guessInput = document.getElementById("guess");
var guess = guessInput.value;

var answer = checkGuess(guess);
alert(answer);
```

获取用户的guess元素，就像上一页一样……

……不过，不再把上一页剩下的所有代码都放在主代码中，我们希望能有一个漂亮的"checkGuess"函数，可以调用它来完成同样的事情。

创建一个checkGuess函数

1. 要创建一个函数，需要用到function关键字，后面是一个函数名，比如"checkGuess"。

2. 为函数提供0个或多个参数。使用参数向函数传值。这里我们只需要一个参数：用户猜到的答案。

```
function checkGuess(guess) {
      var answers = [ "red",
                      "green",
                      "blue"];

      var index = Math.floor(Math.random() * answers.length);

      if (guess == answers[index]) {
          answer = "You're right! I was thinking of " + answers[index];
      } else {
          answer = "Sorry, I was thinking of " + answers[index];
      }
      return answer;
}
```

4. 可以返回一个值作为调用这个函数的结果。这是可选的。在这里，我们返回了一个串，其中包含一个消息。

3. 编写函数体，这要放在大括号之间。体中包含完成函数工作的所有代码。这里的函数体中重用了上一页的代码。

pin51.com 拼吾爱

函数如何工作

所有这些如何工作呢？我们具体调用一个函数时发生了什么？下面做一个深入的分析：

当然了，首先我们需要一个函数。

假设你刚编写了一个新的bark函数，它有两个参数：dogName和dogWeight，另外还包含一点点代码，可以根据狗的重量返回狗叫声。

```
function bark(dogName, dogWeight) {
    if (dogWeight <= 10) {
      return dogName + " says Yip";
    } else {
      return dogName + " says Woof";
    }
}
```

这就是我们简便易行的bark函数。

现在来调用它！

你已经知道如何调用一个函数：只需要使用函数名，并提供所需要的实参。在这里，我们需要两个实参：一个字符串提供狗名，另外还要提供狗的重量，这是一个整数。

下面做出调用，看看它会怎么做：

我们的函数名。

这里传递了两个实参，名字和重量。

```
bark("Fido", 50);
```

"Fido"　　50

```
function bark(dogName, dogWeight) {
    if (dogWeight <= 10) {
      return dogName + " says Yip";
    } else {
      return dogName + " says Woof";
    }
}
```

调用bark时，实参会赋给bark函数中的行参名。

只要行参出现在函数中，就会使用我们传入的值。

www.pin51.com 拼吾爱 最新最全资源分享下载站

另外，让函数体完成工作。

将各个实参值赋给函数中相应的行参后—比如将“Fido”赋给dogName，把整数50赋给dogWeight,接下来就可以执行函数体中的所有语句了。

语句会从上到下执行，就像你原先编写的所有代码一样。不同的是，我们所在的环境中，行参名dogName和dogWeight已经赋值为你传入函数的实参。

```
function bark(dogName, dogWeight) {
    if (dogWeight <= 10) {
        return dogName + " says Yip";
    } else {
        return dogName + " says Woof";
    }
}
```

这里执行体中的所有代码。

可选地，体中可以有return语句……

要记住，函数不一定要返回一个值。不过，在这里，bark函数确实返回了一个值。

…… 通过return语句可以向做出调用的代码返回一个值。下面来看这是如何做的：

通过跟踪函数，可以看到dogWeight不小于或等于10，所以我们使用了else子句，并且作为一个串值返回“Fido says Woof”。

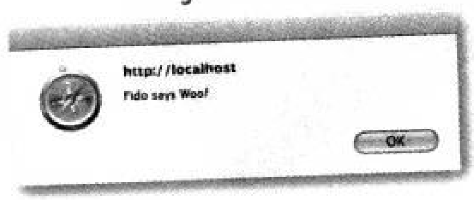
字符串“Fido says Woof”返回到调用代码（也就是调用bark函数的代码）。

返回这个串时，将它赋给变量sound，然后再传递到alert，最后得到对话框。

```
function bark(dogName, dogWeight) {
    if (dogWeight <= 10) {
        return dogName + " says Yip";
    } else {
        return dogName + " says Woof";
    }
}
```

"Fido says Woof"

```
var sound = bark("Fido", 50);
alert(sound);
```





真该好好谈谈了……

知道，我们当然知道，第4章之前你认为现在会飞到HTML5机场，没错，我们到了。不过，在此之前，你确实需要了解HTML5 JavaScript API的一些基础，这一章就来讨论这个内容。

那么这些基础是什么？可以认为HTML5 JavaScript API是由对象、方法（也称为函数）和属性构成的。所以，要想真正掌握这些API，你需要很好地理解这些内容。当然，你也可以试试看，如果不了解这些基础会怎么样。不过倘若是这样，当你不能很好地使用API时（包括犯大量错误，以及编写有bug的代码），可能对API根本不知所措。

所以，在你深入这一章之前，我们只是想提醒你下面要做什么。有一个好消息要告诉你：等你学完这一章，你对于对象、函数和方法以及很多其他相关内容的理解要强过当前98%的JavaScript脚本开发人员。所以一定要用心。



函数闪亮登场

本周访谈：
你不知道的一些东西……

Head First: 欢迎你，函数！我们真希望能深入了解你在做什么。

函数: 很高兴来到这里和大家见面。

Head First: 如今，你们注意到很多刚接触JavaScript的人并不太愿意用你。他们只是一行一行、从上到下编写着自己的代码。他们为什么要了解你呢？

函数: 嗯，真遗憾，要知道我的功能可强大了。可以这样来考虑我：我提供了一个途径，可以写一次代码，然后反复地重用这个代码。

Head First: 既然说到这里，很抱歉我得说两句，如果你只是允许他们反复地做同样的事情……这有些枯燥吧，是不是？

函数: 不，不，绝对不是，函数是参数化的。也就是说，每次你使用函数时，都会传入实参，这样就能根据你传入的内容得到不同的结果。

Head First: 是吗？举个例子？

函数: 假设你要告诉用户他们的购物车中商品总共多少钱，就可以编写一个函数`computeShoppingCartTotal`。然后可以向这个函数传入属于不同用户的不同购物车，这样，每次你就能得到购物车相应的商品金额了。

……另外，再说说你刚才的评价，你说新手不使用函数；这可不是事实，他们一直都在使用函数：`alert`、`document.getElement`、`Math.random`都是函数。他们只是没有定义自己的函数。

Head First: 噢，对了，alert确实是函数。不过另外两个看上去好像不是函数吧。

函数: 哈，它们当然是函数，要知道……先等等……

……噢，刚才有人告诉我读者还不了解这种函数，不过再过几页就会看到了。不管怎么说，函数无处不在。

Head First: 这么说，函数的一个工作就是返回一个值，对吧？我的意思是说，如果我根本没有想要返回的值呢？

函数: 很多函数会返回值，但是函数并不一定非得返回值。很多函数只是做一些工作，比如更新DOM，然后直接返回，而不返回值，这也没问题。

Head First: 那么在这些函数里就不需要return语句了吧？

函数: 说得对。

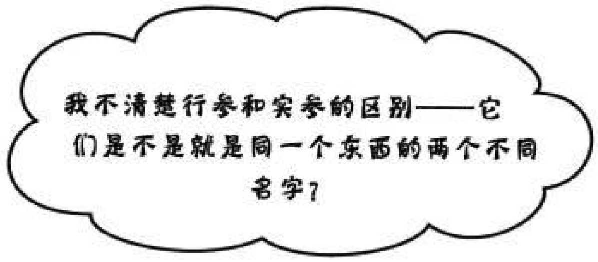
Head First: 嗯，关于命名函数再说两句吧，我听说如果你不想命名，也可以不命名。

函数: 嘿，先别太深奥，把读者吓着了。等他们对我了解多点了再来讨论这个话题怎么样？

Head First: 你可别忘了。

函数: 下面我们来谈谈……

pin51.com 拼吾爱





不，它们完全不同。

定义一个函数时，可以定义它有一个或多个行参。

这里我们定义了3个行参：degrees、mode和duration。

```
function cook(degrees, mode, duration) {
    // your code here
}
```

调用一个函数时，要提供实参：

```
cook(425.0, "bake", 45);
```

这些是实参。这里有3个实参：一个浮点数、一个串和一个整数。

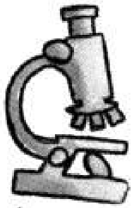
```
cook(350.0, "broil", 10);
```

所以，你只需要定义一次行参，但是调用函数时可以提供很多不同的实参。

你可能会惊奇地发现，居然那么多人都没有搞懂这一点。甚至有些书里都说错了。所以如果你在别处看到过不同的说法，现在应该知道正确的理解是什么……。

定义函数时指定行参，调用函数时提供实参。

拼吾爱



函数剖析

既然你已经知道如何定义和调用一个函数，下面来确保真正掌握这个语法。以下是函数解剖之后的各个部分：

```
function addScore ( level , score ) {
    var bonus = level * score * .1;
    return score + bonus;
}
```

总是以关键字"function"开头。

function关键字后面是你的函数名。

然后是0个或多个用逗号分隔的行参，这些行参放在小括号之间。

即使你的函数没有行参，也需要一对开始和结束小括号，如()。

函数体放在两个大括号之间，包含一组语句（如你常用的那些语句）。

这是函数体的结束大括号。

函数可以包含一个有return关键字的语句，不过这是可选的，没有严格要求。

return语句包含一个表达式，它会作为调用这个函数的结果返回。

there are no Dumb Questions

问： 为什么形参名前面没有var呢？行参就是一个新变量，是吧？

答： 实际上就是这样。函数会为你完成实例化变量的全部工作，所以你不需要在行参名前面加var关键字。

问： 函数名有哪些规则？

答： 对函数命名的规则与变量命名规则是一样的。

问： 我向函数传递了一个变量——如果在函数中改变了相应行参的值，会不会也改变原来的变量？

答： 不会，传递一个基本类型值时，它会复制到行参。我们把这称为"传值"（按值传递）。所以，如果在函数体中改变了行参的值，对原来的实参值没有任何影响。不过传递数组或对象是个例外，稍后就会介绍。

问： 那么，我该怎么在函数中改变一个值呢？

答： 你只能改变全局变量的值（也就是在函数之外定义的变量），或者可以改变你在函数中显式定义的变量。稍后我们就会更详细地讨论这个内容。

问： 如果一个函数没有return语句，它会返回什么？

答： 如果没有一个return语句，这样的函数会返回undefined。

pin51.com

Sharpen your pencil

你已经了解了函数，也知道了向行参传递实参，利用你现有的知识分析下面的代码。跟踪这个代码之后，在下面写出各个变量的值。学习后面的内容之前，对照本章最后的答案看你做得对不对。

```
function dogsAge(age) {
    return age * 7;
}

var myDogsAge = dogsAge(4);

function rectangleArea(width, height) {
    var area = width * height;
    return area;
}

var rectArea = rectangleArea(3, 4);

function addUp(numArray) {
    var total = 0;
    for (var i = 0; i < numArray.length; i++) {
        total += numArray[i];
    }
    return total;
}

var theTotal = addUp([1, 5, 3, 9]);

function getAvatar(points) {
    var avatar;
    if (points < 100) {
        avatar = "Mouse";
    } else if (points > 100 && points < 1000) {
        avatar = "Cat";
    } else {
        avatar = "Ape";
    }
    return avatar;
}
var myAvatar = getAvatar(335);
```

在这里写出各个变量的值……

myDogsAge = ...

rectArea = ...

theTotal = ...

myAvatar = ...



局部变量和全局变量

了解区别或风险

你已经知道，可以在脚本中的任何地方使用var关键字和一个变量名声明一个变量：

```
var avatar;
var levelThreshold = 1000;
```

这些是全局变量；可以在你的JavaScript代码中的任何地方访问。

另外你也见过，还可以在一个函数内部声明变量：

```
function getScore(points) {
    var score;

    for (var i = 0; i < levelThreshold; i++) {
        //code here
    }

    return score;
}
```

points、score和i变量都在函数内部声明。

我们把它们称为局部变量，因为只有函数内部局部知道这些变量。

即使我们在函数中使用了levelThreshold，它仍是一个全局变量，因为它在函数外声明。

如果一个变量在函数外声明，这就是全局变量。如果在函数内部声明，它就是局部变量。

不过，这有什么关系？变量就是变量，不是吗？嗯，是这样的，在哪里声明变量将会确定这个变量对代码中其他部分的可见性，而且，如果你了解这两种变量如何操作，这将有助于你以后编写更可维护的代码（当然，也能帮助你更好地理解别人写的代码）。

了解局部变量和全局变量的作用域

在哪里定义变量将确定它们的作用域。也就是说，哪里定义以及哪里未定义，哪里对代码可见以及哪里不可见。下面来看一个例子，其中同时包含了局部和全局作用域变量。要记住，函数外定义的变量是全局作用域变量，而函数内部定义的变量是局部作用域变量：

```
var avatar = "generic";
var skill = 1.0;
var pointsPerLevel = 1000;
var userPoints = 2008;

function getAvatar(points) {
    var level = points / pointsPerLevel;

    if (level == 0) {
       return "Teddy bear";
    } else if (level == 1) {
       return "Cat";
    } else if (level >= 2) {
       return "Gorilla";
    }
}

function updatePoints(bonus, newPoints) {
    for (var i = 0; i < bonus; i++) {
        newPoints += skill * bonus;
    }
    return newPoints + userPoints;
}

userPoints = updatePoints(2, 100);
avatar = getAvatar(2112);
```

这4个变量都是全局作用域变量。这说明，它们在以下所有代码中定义并且完全可见。

注意，如果页面链接到其他脚本，它们也会看到这些全局变量！

这里的level变量是局部变量，只对getAvatar函数内的代码可见。这说明，只有这个函数能访问level变量。

不要忘了points参数，它的作用域也是局部的（getAvatar函数内部）。

注意getAvatar还使用了pointsPerLevel全局变量。

在updatePoints中有一个局部变量i，i对updatePoints中的所有代码都可见。

bonus和newPoints对于updatePoints也是局部的，不过userPoints是全局的。

在这里，我们只能使用全局变量，不能再访问函数内部的变量，因为他们在全局作用域中不可见。

www.pin51.com 拼吾爱



变量的短暂一生

如果你是一个变量，你得努力工作，而且你的生命可能很短暂。也就是说，除非你是一个全局变量，不过即便是全局变量，生命也是有期限的。但是到底是什么决定了变量的寿命呢？可以这样来考虑：

只要页面存在，全局变量就活着。全局变量从其JavaScript加载到页面开始有生命。不过，一旦这个页面不见了，全局变量的生命也到了尽头。即使你重新加载同样的页面，所有全局变量也已经被破坏，会在新加载的页面中重新创建。

局部变量通常在函数结束时就消失。局部变量会在函数第一次调用时创建，一直存活到函数返回（返回一个值或者不带值返回）。也就是说，可以先得到局部变量的值，在这些变量进入大限之前从函数将它们返回。

这里谈到"通常"，因为还有一些高级的方法可以让局部变量存活更长时间，不过目前先不用考虑这种情况。

这么说，是不是根本无法从页面逃走？如果是一个局部变量，你的生命会像昙花一现，转瞬即逝，而如果你足够幸运，是个全局变量，只要浏览器没有重新加载页面，你就能好好活着。

不过必须有办法逃离页面！我们一定能想到办法！对不对？

可以看看我们的Web存储一章，到时候我们会帮助数据逃离可怕的页面刷新！

pin51.com 拼吾爱 最新最全的资源分享下载站



你会“遮蔽”这个全局变量。

我的意思是：假设你有一个全局变量`beanCounter`，然后声明了一个函数，如下：

```
var beanCounter = 10;

function getNumberOfItems(ordertype) {
    var beanCounter = 0;
    if (ordertype == "order") {
       // do some stuff with beanCounter...
    }
    return beanCounter;
}
```

我们有一个全局变量和一个局部变量。

这样一来，函数中`beanCounter`的所有引用都会指向局部变量，而不是全局变量。所以我们说全局变量被挡在局部变量的阴影里（换句话说，我们无法看到全局变量，因为局部变量挡在前面）。

注意全局变量和局部变量相互之间没有任何影响：如果改变其中一个，它就不会影响到另一个。它们是独立的变量。

www.mn51.com 拼吾爱 最新资源分享

there are no Dumb Questions

问：掌握所有这些局部变量和全局变量的作用域很让人糊涂，那么为什么不都用全局变量呢？我就总是这么做。

答：如果你写的代码很复杂，或者需要在很长一段时间之后维护，你就确实应当好好看一看如何管理变量。如果你过分热衷于创建全局变量，将很难跟踪到变量在哪里使用（以及在哪里对变量值做出了修改），而这可能会导致有bug的代码。与别人合作编写代码时，或者使用了第三方库时，这一点会更为重要（不过，如果这些库编写得很好，则它们应该会有合理的结构来避免这些问题）。

所以说，可以在合适的地方使用全局变量，不过一定要适度，另外要尽可能使用局部变量。等你对JavaScript有了更多经验，就能研究另外一些代码建构技术，使代码更可维护。

如果这是一本深入JavaScript编程的书，则我们还会继续讨论这个主题，不过本书的名字叫《Head First HTML5Programming》，所以建议你继续研究这个主题，来改进你的代码的质量！

问：我的页面中有全局变量，不过我还加载了其他JavaScript文件。这些文件会有不同的全局变量集吗？

答：只有一个全局作用域，所以你加载的各个文件会看到同样的一组变量（并在相同的空间创建全局变量）。正因如此，一定要当心变量的使用，以避免冲突（还要尽可能减少或消除全局变量）。

问：我见过有些代码中，向一个新变量名赋值时并没有使用var关键字。这是怎么回事？

答：嗯，这是可以的。向一个还没有声明的变量名赋值时，会把它当做一个新的全局变量。所以要当心，如果在一个函数内部这样做，你就会创建一个全局变量。提醒一下，我们并不推荐这种编码做法。不仅是因为读代码时可能带来混淆，而且有些人认为以后JavaScript实现可能会改变这种行为（这就有可能破坏你的代码）。

问：使用函数前需要先定义吗？或者它能出现在脚本的任何位置上吗？

答：函数声明可以出现在脚本中的任何位置。只要你愿意，完全可以在使用函数的位置下面声明这个函数。这是可以的，因为首先你要加载页面，浏览器会解析页面中（或外部文件中）的所有JavaScript，开始执行代码前它会看到这个函数声明。还可以把全局变量声明放在脚本中的任何地方，尽管我们建议所有全局变量都在文件的最前面声明，这样更容易找到。

使用多个外部JavaScript文件时有一点要记住，如果不同文件中有两个同名的函数，将使用浏览器最后看到的那个函数。

问：好像所有人都在抱怨JavaScript中滥用全局变量。为什么？是这个语言设计得不好？还是人们不知道他们在做什么？或者是其他什么原因？我们能谈谈吗？

答：JavaScript中经常滥用全局变量。其中部分原因是因为这种语言太容易上手了，让人很容易就开始编写代码是一件好事。因为JavaScript对结构没有做太多要求，也没有给你增加太大负担。不好的一点是，如果这样随意地编写正式代码，过很长时间后必须进行修改和维护（这往往意味着需要调整所有Web页面）。可以这么讲，JavaScript是一个强大的语言，包括诸如对象等很多特性，可以利用这些特性采用一种模块化的方式组织代码。很多书专门讨论这个主题，这一章第二部分（再过几页）会让你对对象有点认识。

pin5i.com 拼吾爱

噢，我们提到过函数也是值吗？

OK，你用变量存储过数值、布尔值、串、数组，诸如此类，不过我们提到过还可以把一个函数赋给变量吗？下面来看看：

```
function addOne(num) {
    return num + 1;
}

var plusOne = addOne;

var result = plusOne(1);
```

定义一个简单的函数，让它的参数加1。

现在来点新鲜的。我们将使用这个函数名addOne，把addOne赋至一个新变量plusOne。

注意这里没有通过addOne()调用函数，我们只是使用了函数名。

plusOne赋为一个函数，所以可以调用它并提供一个整数参数1。

这个调用之后，result等于2。

嗯，之前我们不仅没有提到函数的这个小细节，而且在剖析函数时还有点保留，没有全盘告诉你。实际上，甚至可以不指定函数名。这到底是什么意思？为什么你会想要这样一个东西呢？首先，来看看如何创建一个没有函数名的函数：

```
function(num) {
    return num + 1;
}

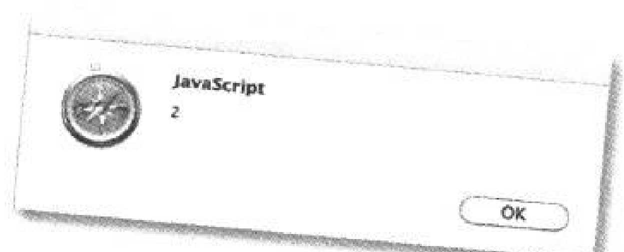
var f = function(num) {
    return num + 1;
}
var result = f(1);
alert(result);
```

这里创建了一个函数，但是没有函数名……嗯……那么怎么用这个函数呢？

同样的，这一次把它赋给一个变量。

然后可以使用这个变量来调用函数。

这个调用之后，result等于2。



www.mn51.com 拼吾爱 最新最全的电子书下载站

BRAIN POWER

来看下面的代码：你认为会发生什么？

有了前面的介绍，这看上去应该好理解一些了……

```
var element = document.getElementById("button");
element.onclick = function () {
    alert("clicked!");
}
```

如果你还没有100%了解也不用担心，稍后就会介绍……

将函数作为值能够做什么？

那么重点是什么？这有什么用？嗯，能够将函数赋给一个变量并不是太重要，我们只是借此向你展示函数实际上是一个值。而且你知道可以把值存储在变量或数组中，可以将值作为参数传递到函数，或者很快就会看到，还可以把值赋给对象的属性。不过，我们不打算继续讨论匿名函数多么有用，将函数作为值有很多用法，下面来看其中一种方法，现在开始有点意思了：

这是一个简单的init函数。

```
function init() {
    alert("you rule!");
}
window.onload = init;
```

这里将我们定义的函数赋给onload处理程序。

看呐，我们将函数用做一个值！

或者甚至还能更有意思一些。

这里我们创建了一个函数，但是没有一个显式的函数名，然后把它的值直接赋给window.onload属性。

如果还不太清楚window.onload，也不用担心，很快就会介绍这个内容。

```
window.onload = function() {
    alert("you rule!");
}
```

哇呜，是不是更简单也更可读了？

你可能已经发现，除了打包代码重用之外，函数还可以做一些有用的事情。为了让你更好地了解如何充分利用函数，下面来学习对象，了解对象在JavaScript中的位置，然后把函数和对象结合起来。

pin51.com 拼吾爱

作者们？喂？你好，我是买HTML5书的那个女孩，还记得我吗？我想知道，所有这些和HTML5到底有什么关系？



嗯，我想我们已经讲过……不过看起来我们已经载你绕了半个城了，里程表一直在跑（原本可以把你一直带到城中心的），不过我们突然想起来下一章要开始深入讨论结合HTML5使用的API。而且，这需要你真正了解函数、对象和其他一些相关的主题。

所以，坚持一下。实际上你已经走了一半了！另外，别忘了，从这一章开始，你会从一个脚本开发人员成长为程序员，从只懂得HTML/CSS变得能够构建真正的应用。

我们已经提到过吧，这可能还能让你挣到更大把的钞票！

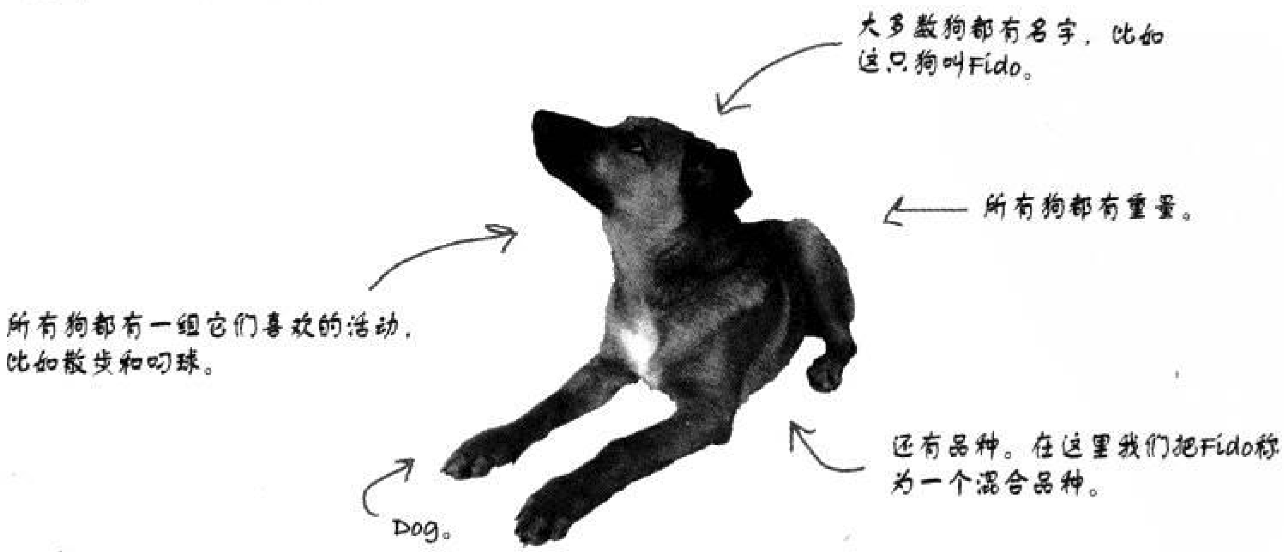
www.pin51.com 拼吾爱



有人谈到“对象”！

嗯，我最喜欢讨论这个话题了！对象可以让你的JavaScript编程技能上升到更高层次。对象是管理复杂代码、理解DOM、组织数据的关键，甚至也是包装HTML5 JavaScript API的基本方法（这还只是列出了对象的几个用途而已）。也就是说，对象是一个很难的内容，是吗？哈！我们就要深入进来了，很快你就会用到对象。

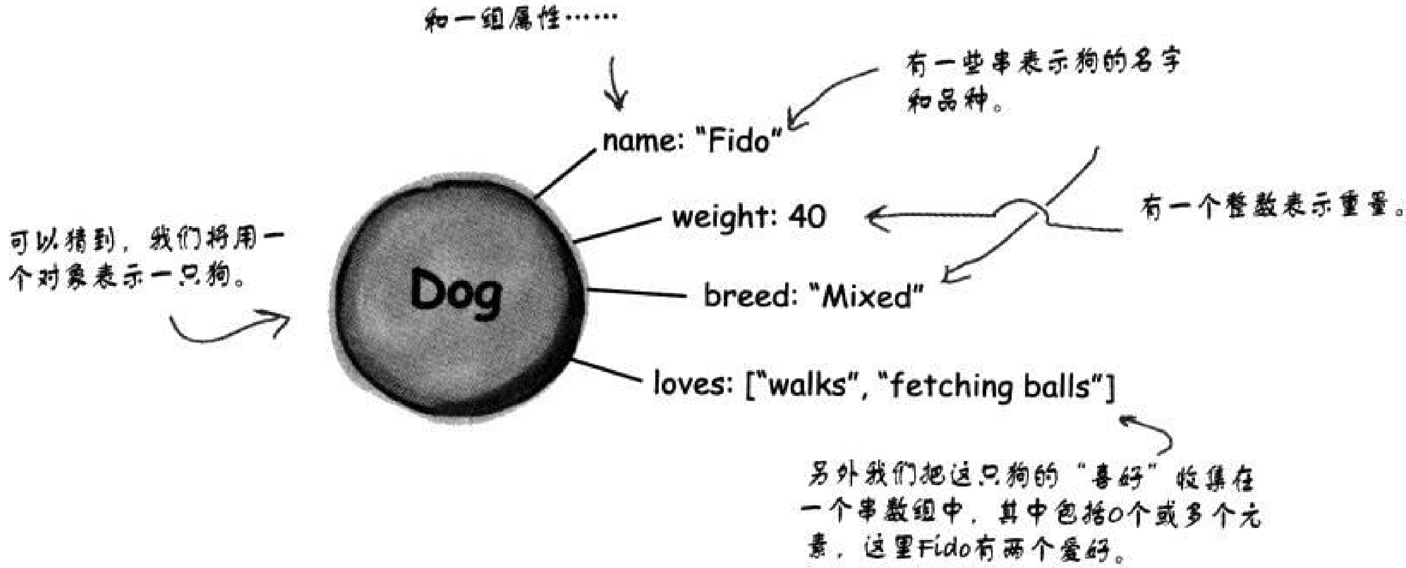
告诉你一个JavaScript对象的秘密：它们实际上就是一个属性集合。举个例子，比如，一只狗。狗是有属性的：



pin51.com 拼吾爱

考虑属性……

当然Fido肯定承认除了一些属性外，它还有很多其他方面，不过对于这个例子，我们需要在软件中掌控的只是这些内容。下面按JavaScript数据类型来考虑这些属性：



如何用JavaScript创建对象

已经有了一个包含一些属性的对象。如何使用JavaScript创建这个对象呢？方法如下：

要把这个对象赋给变量fido。

对象首先是一个左大括号，然后把所有属性放在其中。

注意每个属性用一个逗号分隔。不是分号！

```
var fido = {
    name: "Fido",
    weight: 40,
    breed: "Mixed",
    loves: ["walks", "fetching balls"]
};
```

这个对象有4个属性，name、weight、breed和loves。

注意weight的值是一个数40，breed和name是串。

当然还有一个数组来保存狗的"喜好"。

最新资源下载站 拼吾爱 pin51.com

可以用对象做的一些事情

1 用"点"记法访问对象属性：

```
if (fido.weight > 25) {
    alert("WOOF");
} else {
    alert("yip");
}
```

使用对象以及一个"."和属性名来访问这个属性的值。

使用"."

fido.weight

这里是对象……

……然后是一个属性名。

2 使用一个串结合[]记法访问属性：

```
var breed = fido["breed"];
if (breed == "mixed") {
    alert("Best in show");
}
```

使用对象以及用引号和中括号包围的属性名来访问这个属性的值。

现在属性名两边使用[]。

fido["weight"]

这里是对象……

……属性名用引号括起。

我们发现在这两种记法中，点记法更可读。

3 改变属性的值：

```
fido.weight = 27;
fido.breed = "Chawalla/Great Dane mix";
fido.loves.push("chewing bones");
```

改变Fido的重量……

……它的品种……

……并向它的loves数组增加一个新元素。

直接将一个新元素压入数组末尾。

4 枚举对象的所有属性：

枚举就是完全列出对象的所有属性。

```
var prop;
for (prop in fido) {
    alert("Fido has a " + prop + " property ");
    if (prop == "name") {
        alert("This is " + fido[prop]);
    }
}
```

要枚举属性，我们使用了一个for-in循环。

每次循环迭代时，变量prop会得到下一个属性名的相应串值。

我们使用[]记法来访问这个属性的值。

注意属性的顺序是任意的，所以不能依赖一个特定的顺序。

pin51.com 拼吾爱

5 **处理对象的数组：**

```
var likes = fido.loves;
var likesString = "Fido likes";

for (var i = 0; i < likes.length; i++) {
    likesString += " " + likes[i];
}
alert(likesString);
```

在这里，我们将fido的loves数组赋至变量likes。

可以循环处理likes数组，创建一个likesString，其中包含fido的所有爱好。

可以用这个串做出提示。

6 **向函数传入一个对象：**

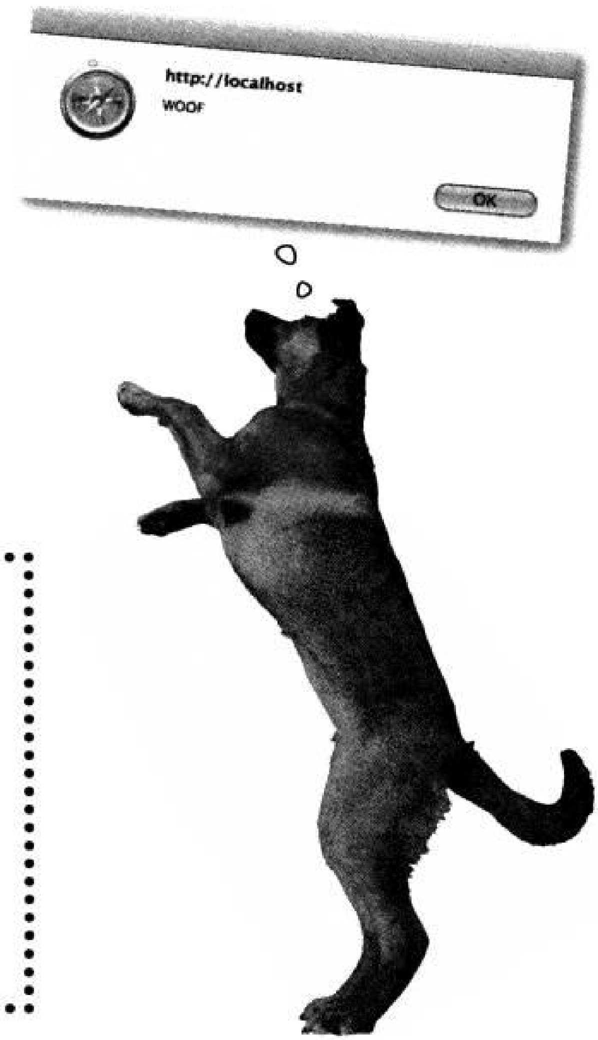
```
function bark(dog) {
   if (dog.weight > 25) {
       alert("WOOF");
   } else {
       alert("yip");
   }
}

bark(fido);
```

可以向函数传入一个对象，就像传入其他变量一样。

在函数中，可以正常地访问这个对象的属性，当然要使用行参名作为对象。

我们将fido作为实参传入函数bark，这个函数需要一个dog对象。

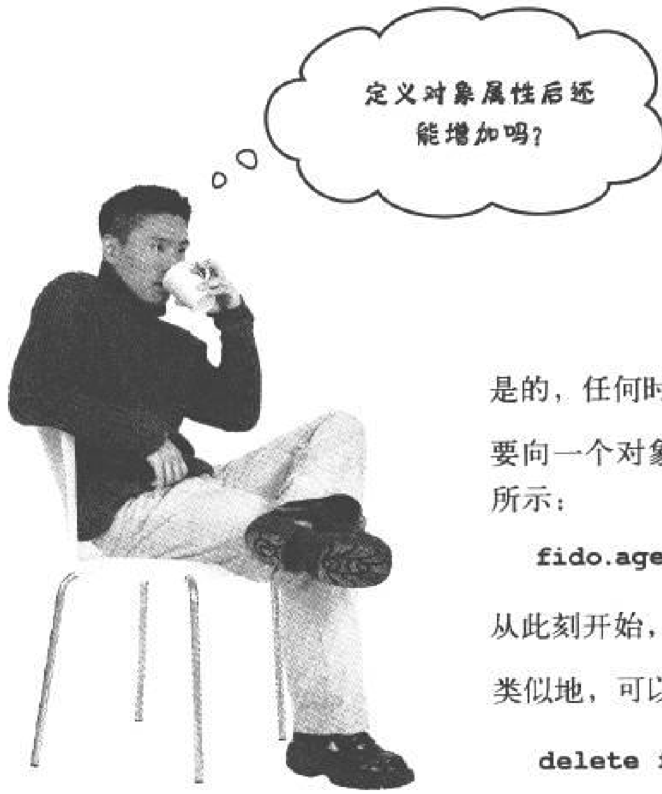


点操作符。

利用点操作符(.)可以访问一个对象的属性。一般来讲，这种记法比[“string”]记法更易读：

- `fido.weight`是fido的重量。
- `fido.breed`是fido的品种。
- `fido.name`是fido的名字。
- `fido.loves`是一个数组，其中包含fido的爱好。

www.java1234.com 拼吾爱 最新编程资源分享下载站



是的，任何时刻都可以增加或删除属性。

要向一个对象增加属性，只需要为一个新属性赋一个值，如下所示：

```
fido.age = 5;
```

从此刻开始，fido就有了一个新属性：age。

类似地，可以用delete关键字删除任何属性，如下：

```
delete fido.age;
```

删除一个属性时，并不只是删除这个属性的值，你会删除属性本身。实际上，如果删除了fido.age之后再使用这个属性，会计算为undefined。

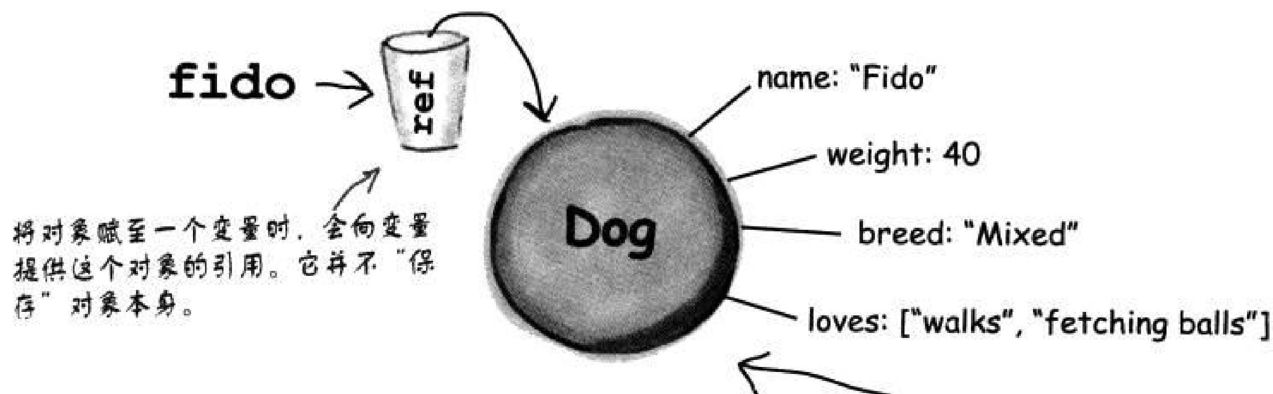
如果属性成功删除（或者如果你要删除一个不存在的属性，也可能你想删除的根本不是一个对象的属性），delete表达式会返回true。

pin51.com 拼吾爱

谈谈向函数传入对象

我们已经谈到过如何将参数传递到函数。实参会按值传入，所以如果传递一个整数，相应的函数行参会得到这个整数值的一个副本，并在函数中使用。对象也是如此，不过，首先还需要进一步分析为变量赋一个对象时变量中包含什么，才能真正了解传递对象的含义。

将一个对象赋至变量时，这个变量会包含这个对象的一个引用，而不是对象本身。可以把引用想成是对象的一个指针。



所以，调用一个函数并传入一个对象时，实际上只传递了对象引用——不是对象本身，而只是它的一个"指针"。这个引用的副本会传递到行参，它指向原来的对象。



嗯，这些到底是什么意思？改变对象的一个属性时，你修改的是原对象的属性，而不是副本，所以不仅可以在函数内部看到对对象的所有改变，而且在函数外一样能看到所有改变。下面一步一步跟踪一个例子，这里对dog使用一个`loseWeight`函数……

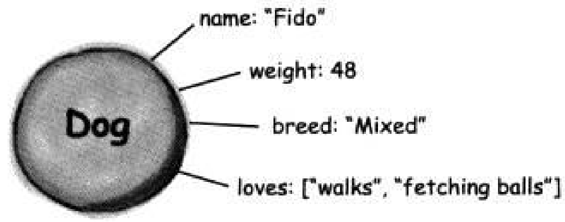
在幕后

让Fido节食……

下面来看把fido传入loseWeight并改变dog.weight属性会发生什么。

① 我们定义了一个对象fido，并把这个对象传入一个函数loseWeight。

fido是一个对象的引用，这说明对象并不放在fido变量中，只是由fido变量指向这个对象而已。



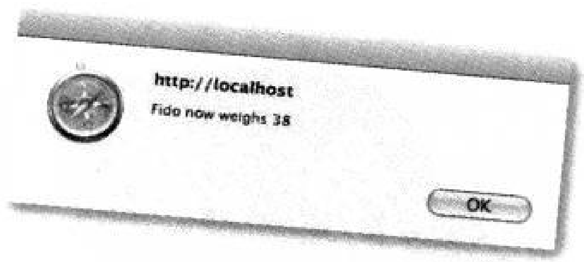

```
fido.weight = 48;
 ...
loseWeight(fido);
```

将fido传入一个函数时，会传递这个对象的引用。

② loseWeight函数的dog形参得到fido引用的一个副本。所以，对形参属性的任何改变都会影响传入的对象。

把fido传入loseWeight时，赋给dog形参的是引用的一个副本，而不是对象的副本。所以fido和dog指向同一个对象。

dog引用是fido引用的一个副本。




```
function loseWeight(dog) {
  dog.weight = dog.weight - 10;
}

alert(fido.name + " now weighs " + fido.weight);
```

所以，将dog.weight减去10磅时，也改变了fido.weight的值。

NOW SHOWING AT THE WEBVILLE CINEMA



Web镇影院希望为他们的JavaScript API提供帮助；我们从简单的开始，先为他们设计movie对象。我们需要两个movie对象，分别包含影片名、类别、影片等级（1~5星）和一组放映时间。在这里大致画出你的movie对象设计（可以使用前面的dog对象作为模板）。下面是一些示例数据，可以用来填充你的对象。

“Plan 9 from Outer Space”，3:00pm、7:00pm 和11:00pm分别放映一场；所属类别为“恐怖片”；等级为2星。

“Forbidden Planet”，5:00pm和9:00pm放映；所属类别为“科幻片”；等级为5星。

答案就在下一页，不过完成这个练习前先不要看答案。真的，一定要做到。

完全可以增加你自己喜欢的影片。

在这里设计你的对象。



NOW SHOWING AT THE WEBVILLE CINEMA SOLUTION

如何创建你的movie对象?

下面是我们的答案:

我们创建了两个对象：movie1和movie2，分别表示两部影片。

movie1有4个属性：片名、类别、等级和放映时间。

```
var movie1 = {
    title: "Plan 9 from Outer Space",
    genre: "Cult Classic",
    rating: 5,
    showtimes: ["3:00pm", "7:00pm", "11:00pm"]
};
```

片名（title）和类别（genre）是串。

等级（rating）是一个数字。

放映时间（showtimes）是一个数组，包含影片的放映时间（分别作为串）。

movie2也有4个属性：片名、类别、等级和放映时间。

```
var movie2 = {
    title: "Forbidden Planet",
    genre: "Classic Sci-fi",
    rating: 5,
    showtimes: ["5:00pm", "9:00pm"]
};
```

记住要用逗号分隔属性。

这里使用与movie1同样的属性名，但有不同的属性值。



www.pin51.com 制吾爱 最新编程

下一次放映时间是……

对于如何结合对象和函数我们已经有点认识了。下面更进一步，来编写一些代码指出影片的下一次放映时间。这个函数取一个影片作为参数，返回一个串，其中包含根据你的当前时间确定的下一次放映时间。



以下是我们的新函数，它取一个movie对象。

我们使用JavaScript的Date对象获取当前时间。先不用担心这个对象的细节，只需要知道它能返回当前时间（单位为毫秒）。

现在使用movie的数组showtimes，迭代处理各个放映时间。

对于每个放映时间，得到相应的时间值（单位为毫秒），然后比较。

如果时间还没有到，就是下一次放映时间，所以返回这个时间。

如果没有放映场次了，就返回null。

```
function getNextShowing(movie) {
    var now = new Date().getTime();

    for (var i = 0; i < movie.showtimes.length; i++) {
        var showtime = getTimeFromString(movie.showtimes[i]);
        if ((showtime - now) > 0) {
            return "Next showing of " + movie.title + " is " + movie.showtimes[i];
        }
    }
    return null;
}
```

成品代码

下面给出一些成品代码，它取一个串，格式类似1am或3pm，然后把这个串转换为单位为毫秒的时间。

```
function getTimeFromString(timeString) {
    var theTime = new Date();
    var time = timeString.match(/(\d+)(?::(\d\d))?\s*(p?)/);
    theTime.setHours( parseInt(time[1]) + (time[3] ? 12 : 0) );
    theTime.setMinutes( parseInt(time[2]) || 0 );
    return theTime.getTime();
}
```

先不用担心这个代码。这里使用了正则表达式，后面学习JavaScript时你会学到的。对现在来说，可以先直接使用这个代码！

现在使用这个函数，调用getNextShowing，并在提醒中使用这个函数返回的串。

再对movie2做同样的处理。

```
var nextShowing = getNextShowing(movie1);
alert(nextShowing);
nextShowing = getNextShowing(movie2);
alert(nextShowing);
```

如何"串链"……

在前面的代码中你注意到这个了吗？

```
movie.showtimes.length
```

看起来这和我们以前看到的好像不一样。这实际上是一系列步骤的简写，原来我们需要用一系列步骤从movie对象得到showtimes数组的长度。我们原本可以这样写：

```
var showtimesArray = movie.showtimes;
var len = showtimesArray.length;
```

首先获取showtimes数组。

然后用它访问length属性。

不过，把表达式串链在一起就可以一步到位。下面逐步分析这是如何做到的：

```
movie.showtimes.length
```

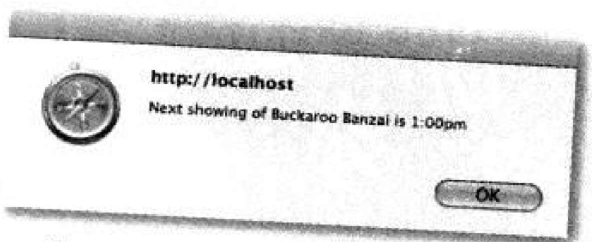
1. 得到movie对象。
2. 它有一个showtimes属性，这是一个数组。
3. 数组有一个属性length。

试一试

输入上一页的代码，试着来运行。你会看到getNextShowing函数会根据传入的影片，确定它的下一次放映时间。完全可以创建一些你自己的新movie对象，运行试试看。我们就创建了一个新影片，在当地时间12:30pm运行这个代码：

```
var banzaiMovie = {
    title: "Buckaroo Banzai",
    genre: "Cult classic",
    rating: 5,
    showtimes: ["1:00pm", "5:00pm", "7:00pm"]
}

var nextShowing = getNextShowing(banzaiMovie);
alert(nextShowing);
```



注意：我们的代码可没有达到"生产代码"的质量；如果在最后一次放映时间之后运行这个代码，就会得到null。你可以第二天再试试。☺

pin51.com 拼吾爱 最新下载站

对象也可以有行为……

你不会认为对象只能存储数字、串和数组吧？对象是有生命的，它们还能做事情。狗不会总坐在那里，它们会叫、会跑、会玩接东西，dog对象也一样！根据这一章学到的知识，你现在完全可以为对象增加行为了。下面来说明如何实现：

对象中有函数时，我们说这个对象有一个方法。

```
var fido = {
    name: "Fido",
    weight: 40,
    breed: "Mixed",
    loves: ["walks", "fetching balls"]
    bark: function() {
        alert("Woof woof!");
    }
};
```

可以像这样直接向对象增加一个函数。

我们不说“对象中的函数”，而会说这是一个方法。它们是一样的，不过所有人都称对象函数为方法。

注意，我们使用了匿名函数，并把它赋给对象的bark属性。

要调用对象的一个方法，可以采用点记法使用对象名和方法，并提供所需的所有实参。

```
fido.bark();
```

通过在对象上调用方法，让对象做事情。在这里，我们调用了fido的bark方法。





再回到Web镇影院……

既然你已经掌握了有关对象的更多知识，下面可以再回来改进我们的影院代码。我们已经编写了一个getNextShowing函数，它取一个movie作为参数，不过可以把它创建为一个方法，使它成为movie对象的一部分。下面就来做这个工作：

```
var movie1 = {
    title: "Plan 9 from Outer Space",
    genre: "Cult Classic",
    rating: 5,
    showtimes: ["3:00pm", "7:00pm", "11:00pm"],

    getNextShowing: function(movie) {
        var now = new Date().getTime();

        for (var i = 0; i < movie.showtimes.length; i++) {
            var showtime = getTimeFromString(movie.showtimes[i]);
            if ((showtime - now) > 0) {
                return "Next showing of " + movie.title + " is " + movie.showtimes[i];
            }
        }
        return null;
    }
};
```

把我们的代码放在movie1对象的一个方法中，属性名为getNextShowing。

不过我们知道这可能不太对……

实际上我们不能直接把函数放在这个对象中，因为getNextShowing要取一个movie参数，而我们真正想要的是像下面这样调用getNextShowing：

```
var nextShowing = movie1.getNextShowing();
```

这里不应有参数，我们显然知道希望得到哪个影片的下一次放映时间，也就是说，我们想知道movie1的下一次放映时间。

那么，如何修正这个问题呢？必须去除getNextShowing方法定义中的行参，不过这样一来，就需要对代码中的所有movie.showtimes引用做些处理，这是因为，一旦删除了这个行参，movie就不再作为一个变量存在了。下面来看一看……

pin51.com 拼吾爱 最新最全资源分享下载站

下面去掉movie行参……

我们已经删除了movie行参以及它的所有引用。这会得到下面的代码：

```
var movie1 = {
    title: "Plan 9 from Outer Space",
    genre: "Cult Classic",
    rating: 5,
    showtimes: ["3:00pm", "7:00pm", "11:00pm"],

    getNextShowing: function() {
       var now = new Date().getTime();

       for (var i = 0; i < showtimes.length; i++) {
          var showtime = getTimeFromString(showtimes[i]);
          if ((showtime - now) > 0) {
              return "Next showing of " + title + " is " + showtimes[i];
          }
       }
       return null;
    }
};
```

我们突出显示了下面的变化……

这看起来很合理，不过我们需要再仔细考虑getNextShowing方法如何使用属性showtimes……

……我们常用局部变量（showtimes不是局部变量）和全局变量（showtimes也不是全局变量）。唉……

噢，还有一个小属性。

现在怎么办？

嗯，现在遇到难题了：我们要得到属性showtimes和title的引用。通常情况下，在一个函数中我们会引用局部变量、全局变量或者函数的一个形参，不过showtimes和title是movie1对象的属性。嗯，这可能也行……看起来JavaScript应该足够聪明，能发现这一点吧？

不，这是不行的。你可以自己试一试。JavaScript会告诉你showtimes和title变量是未定义的。怎么回事？

是这样的：这些变量是一个对象的属性，但是我们没有告诉JavaScript它是哪一个对象。你可以自言自语，“嗯，显然我们是指这个（THIS）对象，就是这一个！这还有什么迷惑的呢？”不错，我们确实就是要得到这个对象的属性。实际上，JavaScript中就有一个名为this的关键字，你要通过它告诉JavaScript你是指目前所在的这个对象。

现在，情况确实比原先看起来复杂一些了,稍后我们就会解决这个问题，不过，对于现在，我们先增加这个this关键字，让代码运转起来。

拼吾爱

增加"this"关键字

下面在每一处指定属性的地方增加`this`，告诉JavaScript我们希望得到这个对象中的属性：

```
var movie1 = {
    title: "Plan 9 from Outer Space",
    genre: "Cult Classic",
    rating: 5,
    showtimes: ["3:00pm", "7:00pm", "11:00pm"],

    getNextShowing: function() {
        var now = new Date().getTime();

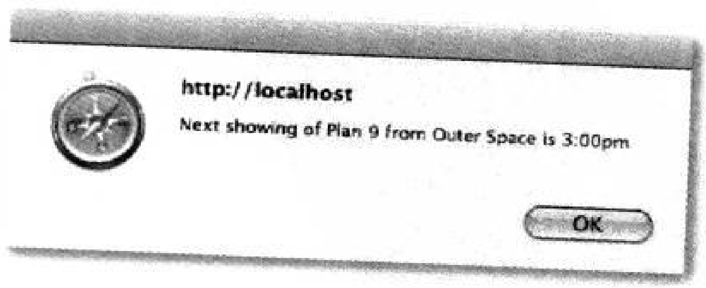
        for (var i = 0; i < this.showtimes.length; i++) {
            var showtime = getTimeFromString(this.showtimes[i]);
            if ((showtime - now) > 0) {
                return "Next showing of " + this.title + " is " + this.showtimes[i];
            }
        }
        return null;
    }
};
```

这里在每个属性前面增加一个this关键字，指示我们想得到movie1对象的引用。

试一试"this"



输入上面的代码，并把`getNextShowing`函数增加到`movie2`对象（只需复制粘贴）。然后对之前的测试代码做下面的修改。再运行这个代码！我们得到下面的结果：




```
var nextShowing = movie1.getNextShowing();
alert(nextShowing);
nextShowing = movie2.getNextShowing();
alert(nextShowing);
```

注意，现在我们在这个对象上调用getNextShowing。这样更明确，不是吗？

pin51.com 拼吾爱





哈，好眼力。

如果你发现我们把`getNextShowing`复制到多个movie对象时是在复制代码，说明你很敏锐，很有直觉。“面向对象”编程的一大目标就是最大化代码重用。在这里我们没有重用任何代码，实际上，我们创建的每个对象都是一次性的，而且我们的movie对象按约定（另外由于复制和粘贴）刚好是一样的。这不仅是浪费，还很容易出错。

还有一种更好的办法，可以使用一个构造函数。什么是构造函数？它就是一个特殊的函数，接下来我们就会写这样一个函数，它能为我们创建对象，并让所有对象都一样。可以把它想象成一个小工厂，根据你希望的属性值设置对象，然后给你奉上一个漂亮的新对象，其中包含所有正确的属性和方法。

下面来创建一个构造函数……

www.aibbt.com 最新最全的电子书下载站 拼吾爱

如何创建构造函数

下面为狗建立一个构造函数。我们已经知道希望有怎样的dog对象：它们有名字、品种和重量属性，另外还有一个吠叫方法。所以构造函数的工作就是得到这些属性值作为参数，然后返回一个“活灵活现”时刻准备吠叫的dog对象。代码如下：

构造函数看上去与常规函数很类似。不过根据约定，命名这个函数时首字母要大写。

构造函数的行参取我们希望的对象属性值。

```
function Dog(name, breed, weight) {
    this.name = name;
    this.breed = breed;
    this.weight = weight;
    this.bark = function() {
        if (this.weight > 25) {
            alert(this.name + " says Woof!");
        } else {
            alert(this.name + " says Yip!");
        }
    };
}
```

在这里，我们把对象属性初始化为传入构造函数的值。

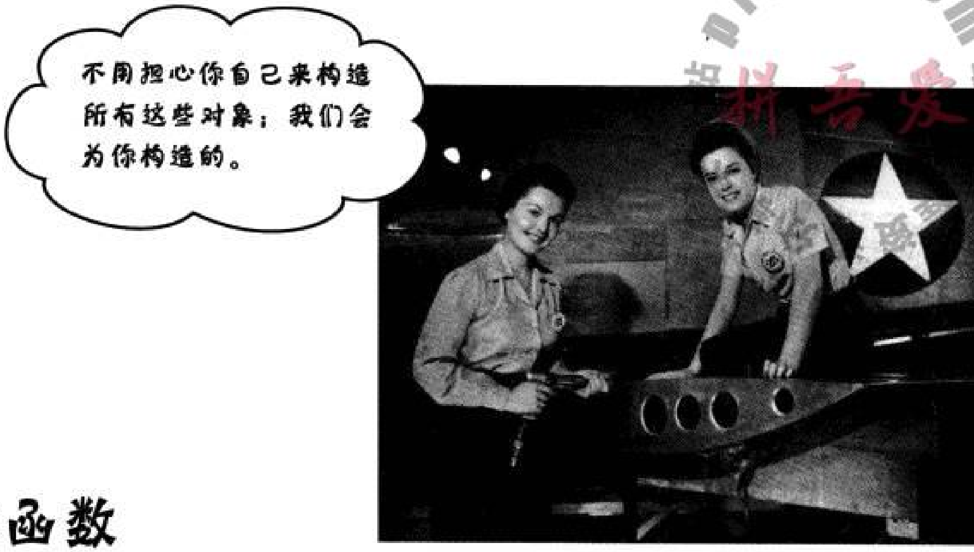
属性名和行参名不要求相同，不过它们通常是一样的，这也是一个约定。

可以在构造的对象中包含bark方法，将bark属性初始化为一个函数值，这类似于之前的做法。

方法中需要使用“this.weight”和“this.name”来指示对象中的属性，这与前面一样。

注意这个语法与对象语法有所不同。这些是语句，所以需要用一个“;”结束各语句，就像我们通常在函数中的做法一样。

下面再把整个过程整理一遍，确保我们已经掌握了有关内容。Dog是一个构造函数，它取一组实参，这正是我们希望的属性初始值：名字、品种和重量。一旦有了这些值，再使用this关键字为属性赋值。它还定义了bark方法。所有这些工作的结果是什么？Dog构造函数会返回一个新对象。下面来看如何具体使用构造函数。



现在使用我们的构造函数

既然构建了我们的工厂，现在则可以用它来创建一些狗。还有一点没有告诉你，要用一种特殊的方式调用构造函数，在调用前面加上关键字new。下面给出几个例子：

要创建一只狗，要结合构造函数使用new关键字。

然后像任何其他函数一样调用。

```
var fido = new Dog("Fido", "Mixed", 38);
var tiny = new Dog("Tiny", "Chawalla", 8);
var clifford = new Dog("Clifford", "Bloodhound", 65);

fido.bark();
tiny.bark();
clifford.bark();
```

我们创建了3个不同的Dog对象，分别传入不同的实参来定制各个狗。

一旦得到对象，就可以调用它们的bark方法让每个Dog吠叫。



下面再来回顾这里发生了什么：我们创建了3个不同的Dog对象，每个对象都有自己的属性，这里结合前面创建的Dog构造函数还使用了new关键字。构造函数返回一个Dog对象，并用我们传入的实参定制。

接下来，对每个对象调用bark方法，注意所有狗都共享同一个bark方法，每个狗吠叫时，this指向做出调用的那个dog对象。所以如果对fido调用bark方法，那么，在bark方法中，this就设置为fido对象。下面将更深入地分析这是如何做到的。

mn51.com 最新最全资源 拼吾爱

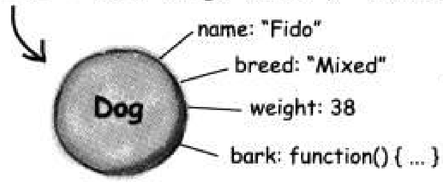
在幕后

到底是怎么回事？

只要在一个方法代码中放入this，就会把它解释为调用这个方法的对象的一个引用。所以，如果调用fido.bark，则this会引用fido。或者，如果在Dog对象tiny上调用这个方法，那么在这个方法中this就会引用tiny。this怎么知道它表示哪个对象呢？下面就来看一看：

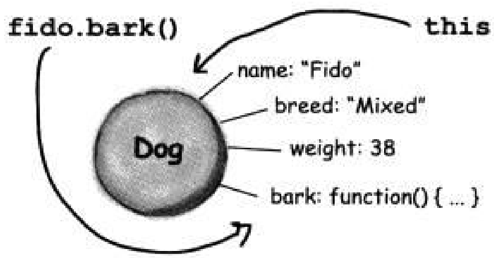
① 假设将一个dog对象赋至fido：

```
fido = new Dog("Fido", "Mixed", 38);
```



这是新的Dog对象，用我们希望的属性值实例化。

② 在fido上调用bark()：

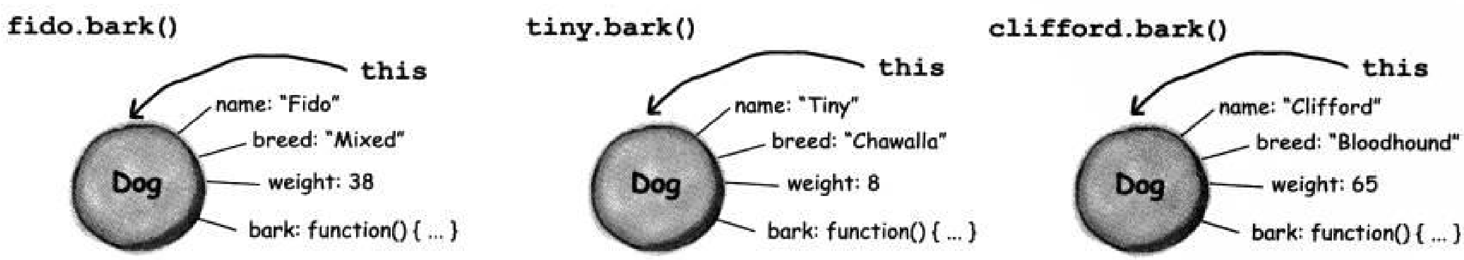


在一个对象上调用某个方法时，JavaScript会设置this指向这个对象本身。所以在这里，this指向fido。

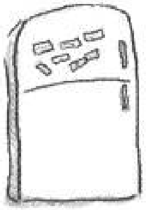
所以引用this.name时，我们知道名字应该是"Fido"。

③ 所以"this"总是指向调用方法的当前对象，而不论我们创建了多少个可以吠叫的狗：

可以在任何dog对象上调用bark，执行方法体的代码之前this会赋为当前的特定dog。



pin51.com 拼吾爱



代码磁贴

冰箱上有一个能正常工作的Movie构造函数，不过有些磁贴掉到地上了。你能帮忙把它整理好吗？要当心，地板上原来就有一些磁贴，可能会干扰你。

```
function ____________(__________, __________, rating, showtimes) {
    this.title = __________;
    this.genre = genre;
    this.____________ = rating;
    this.showtimes = ____________;
    this.getNextShowing = function() {
        var now = new Date().getTime();
        for (var i = 0; i < ______________.length; i++) {
            var showtime = getTimeFromString(this.______________[i]);
            if ((showtime - now) > 0) {
                return "Next showing of " + __________ + " is " + this.showtimes[i];
            }
        }
    } ______
}
```

使用这些磁贴完成代码。

title

Movie

Woof

rating

function

showtimes

this.showtimes

;

bark()

this.title

,

genre

this

www.pin51.com 拼吾爱 最新最全的资源分享下载站

there are no Dumb Questions

问：函数和方法真正的差别是什么？毕竟，如果它们是同一个东西，为什么还要叫不同的名字呢？

答： 根据约定，如果对象有一个函数，我们就称它是一个方法。它们都以同样的方式工作，只不过调用对象的方法时要使用点操作符，而且方法可以使用this来访问调用这个方法的对象。可以认为函数是一段能调用的独立代码，而方法是与一个特定对象关联的行为。

问：这么说，如果用构造函数创建对象，而且这些对象有一个方法，那么所有这些对象都共享这个方法的相同代码啰？

答： 没错，这正是面向对象编程的优点之一：你可以在一个地方为一类对象（比如所有dog对象）创建代码，所有dog对象都会共享这个代码。要让各个狗有所不同，现在的做法是利用属性，并使用this访问这些属性。

问：我能把this设置为我选择的一个值吗？如果可以，会不会弄乱了？

答： 不行，不能把this设置为任何值。要记住，this是一个关键字，而不是一个变量！看起来它像是一个变量，表现也有些类似，不过它确实不是变量。

问：this在对象方法之外有值吗？

答： 没有，如果没有调用一个对象方法，this就是未定义的（undefined）。

问：所以应该这么考虑this：在一个对象上调用一个方法时，执行这个方法的整个过程中this的值都设置为这个对象，这样理解对吗？

答： 在对象的体中确实是这样，this就是对象本身。不过有些特殊的情况下并非如此；例如，如果对象中还包括对象，情况就会更为复杂。如果开始在对象中嵌套对象，就需要仔细查看语义，不过以上理解是一个很好的一般原则。

问：我听说面向对象编程中可以有对象类，它们相互继承。比方说，可以有一个哺乳动物类，狗和猫都可以继承这个哺乳动物类。在JavaScript能这么做吗？

答： 当然能。JavaScript使用了一种称为原型继承的机制，与严格的基于类的模型相比，这是一种更为强大的模型。深入介绍原型继承超出了本书的范畴，不过，不管怎样，这会让我们更有信心编写JavaScript代码。

问：我们说new Date()时就是在使用一个构造函数，是这样吧？

答： 对，问得好！Date是JavaScript中的一个内置构造函数。如果调用new Date()，你就会得到一个Date对象，这个对象包含很多有用的方法，可以用来处理日期。

问：我们自己编写的对象与用构造函数创建的对象之间有什么区别吗？

答： 主要区别在于你如何创建对象。用大括号和逗号分隔的属性所写的对象称为“对象字面量”。你是照字面把这些对象输入到代码中！如果想要一个类似的对象，就必须自行输入，确保它有同样的属性。由构造函数创建的对象则是通过使用new和一个构造函数创建的，它会返回对象。你可以使用构造函数创建多个有相同属性的对象，不过如果愿意，这些属性就可以有不同的属性值。

pin51.com 拼吾爱 最新最全资源分享下载站



代码磁贴

冰箱上有一个能正常工作的Movie构造函数，不过有些磁贴掉到地上了。你能帮忙把它整理好吗？要当心，地板上原来就有一些磁贴，可能会干扰你。

这是一个构造函数，所以函数名要使用“Movie”。

```
function Movie(title, genre, rating, showtimes) {
    this.title = title;
    this.genre = genre;
    this.rating = rating;
    this.showtimes = showtimes;
    this.getNextShowing = function() {
        var now = new Date().getTime();
        for (var i = 0; i < this.showtimes.length; i++) {
            var showtime = getTimeFromString(this.showtimes[i]);
            if ((showtime - now) > 0) {
                return "Next showing of " + this.title + " is " + this.showtimes[i];
            }
        }
    };
}
```

为想要定制的属性传入值：title、genre、rating和showtimes……

……并初始化这些属性。

要在对象中引用属性，需要使用this关键字。

不要忘了用一个分号结束这个语句！

多余的磁贴。

function　this.showtimes　Woof　bark()　,　this

试一试你的构造函数

既然有了一个Movie构造函数，现在就来创建一些Movie对象！输入Movie构造函数，然后增加下面的代码，试着运行你的构造函数。我们相信，你肯定会同意这是一种更为简单的创建对象的方法。

```
var banzaiMovie = new Movie("Buckaroo Banzai",
                            "Cult Classic",
                            5,
                            ["1:00pm", "5:00pm", "7:00pm", "11:00pm"]);

var plan9Movie = new Movie("Plan 9 from Outer Space",
                           "Cult Classic",
                           2,
                           ["3:00pm", "7:00pm", "11:00pm"]);

var forbiddenPlanetMovie = new Movie("Forbidden Planet",
                                     "Classic Sci-fi",
                                     5,
                                     ["5:00pm", "9:00pm"]);

alert(banzaiMovie.getNextShowing());
alert(plan9Movie.getNextShowing());
alert(forbiddenPlanetMovie.getNextShowing());
```

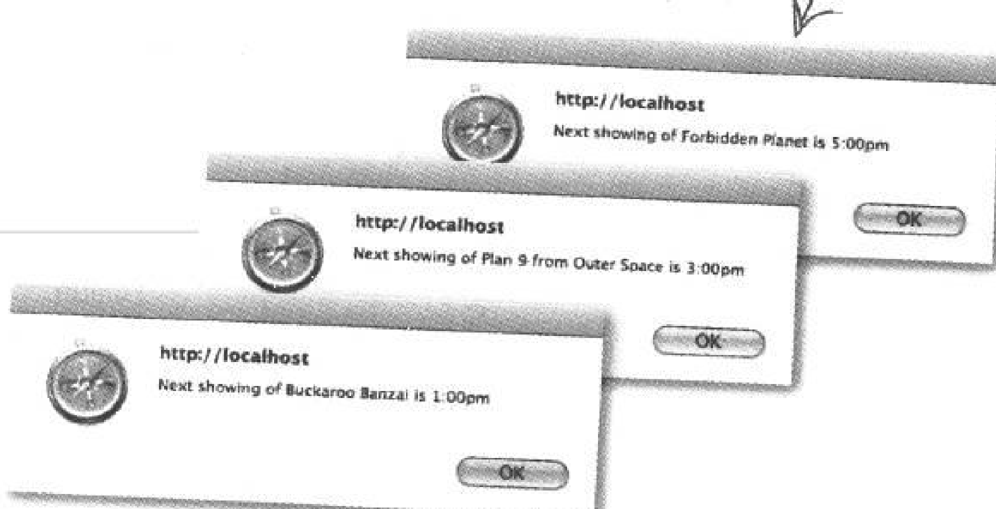
首先为电影"Buckaroo Banzai"创建一个movie对象（这是我们最喜欢的恐怖片之一）。将这些值输入行参。

注意可以把showtimes的数组值直接放在函数调用中。

接下来是影片"Plan 9 from Outer Space"……

当然，然后是"Forbidden Planet"。

一旦创建了所有对象，下面可以调用getNextShowing方法，提醒用户下一个放映时间。



pin51.com 拼吾爱

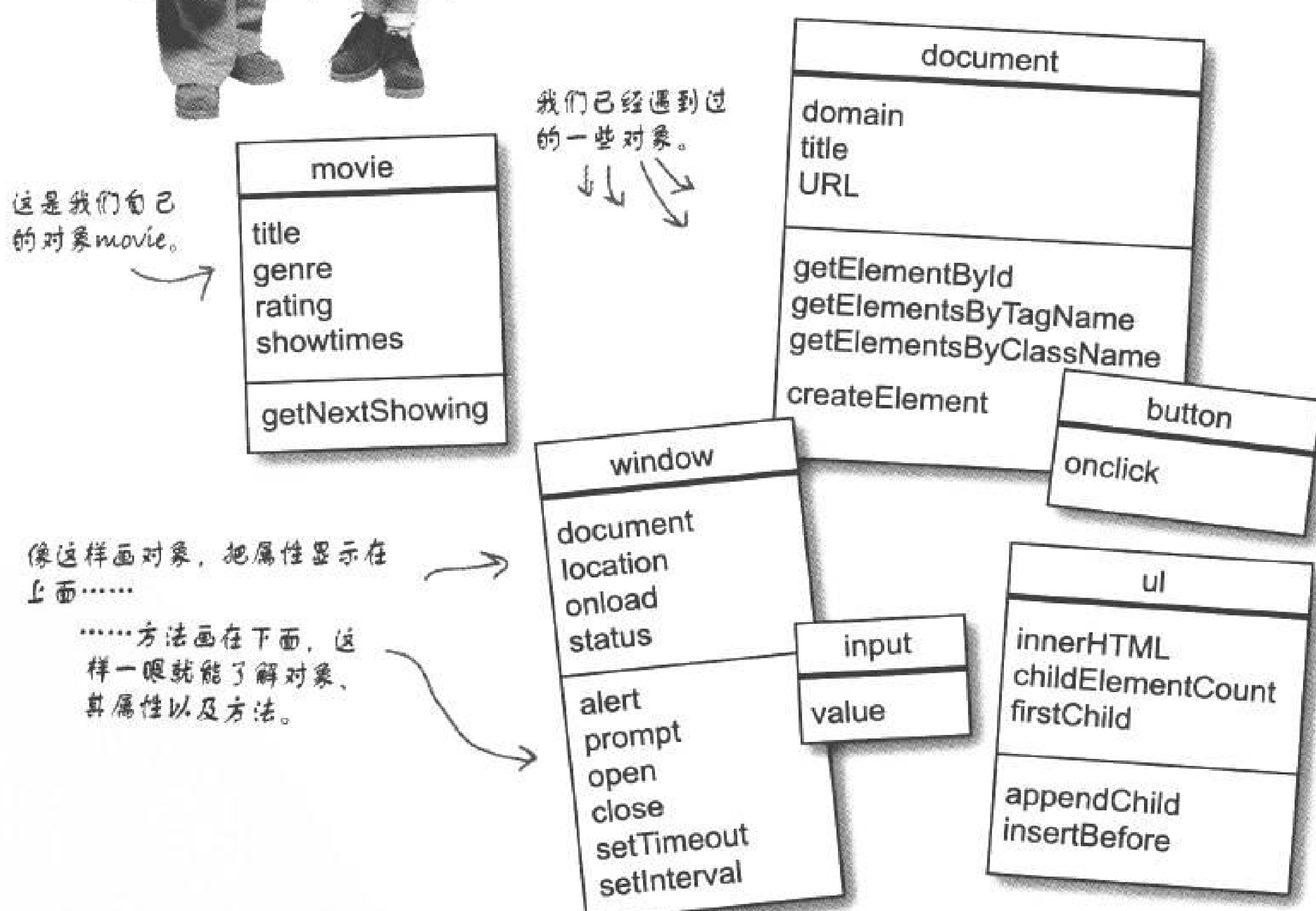


祝贺你，你已经学完了函数和对象！既然已经全面了解，结束这一章之前，下面花点时间在真正的环境中对JavaScript对象做个检查。也就是说，在它们的原栖息地：浏览器！

现在，你可能已经注意到……

……你的身边到处都是对象。例如，document和window就是对象，我们从document.getElementById返回的元素也是对象，我们还会遇到很多对象，这只是其中的几个，等我们谈到HTML5 API，就会看到对象无处不在！

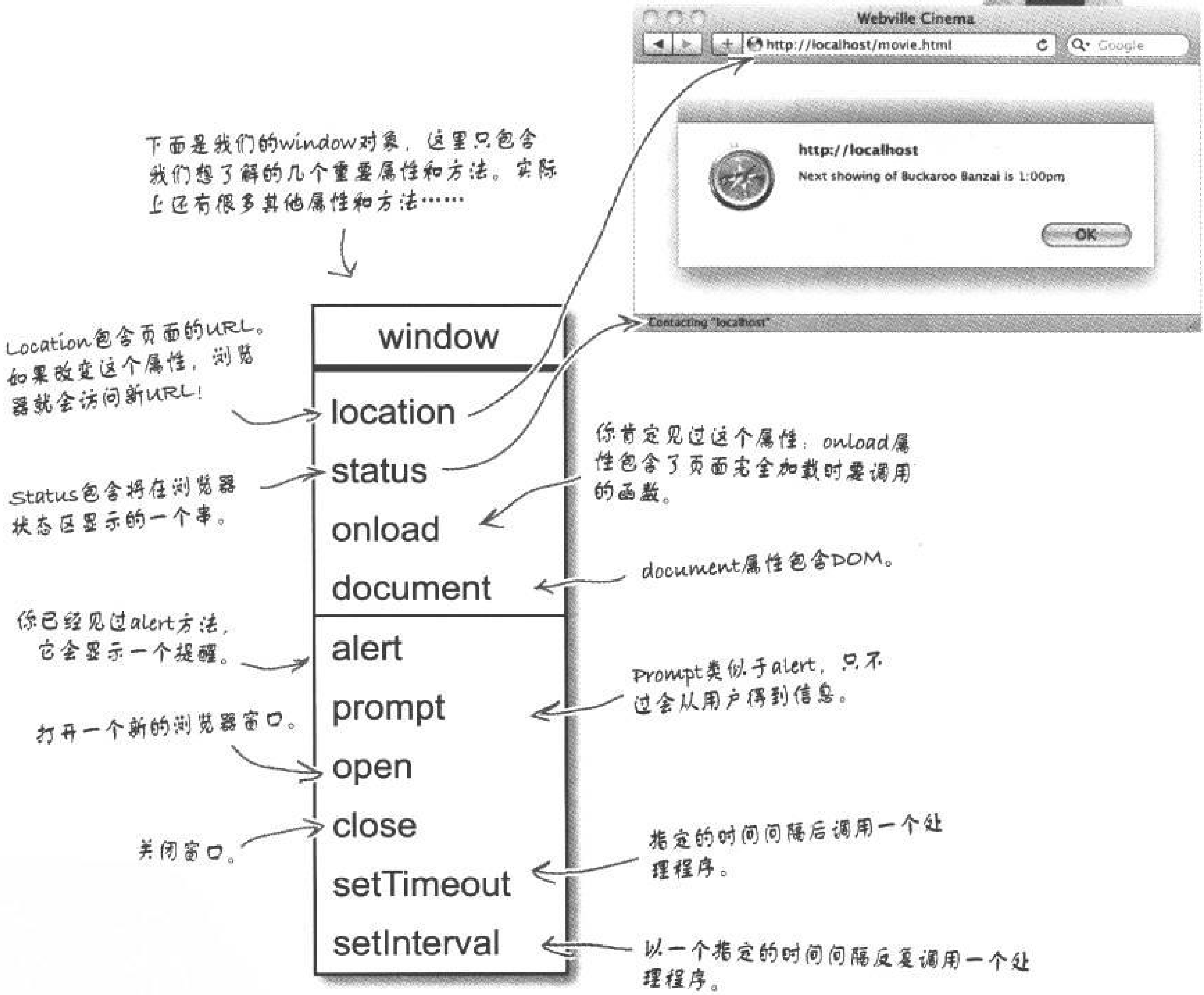
下面再来看看这本书前面一直用到的一些对象：





window对象到底是什么？

为浏览器编写代码时，你的生活中总少不了window对象。window对象表示你的JavaScript程序的全局环境，同时还表示应用的主窗口，因此，其中包含很多核心的属性和方法。下面来看看这个对象：



pin51.com 拼吾爱



window是个全局对象。

看起来可能有点奇怪，不过window对象相当于你的全局环境，所以即使没有在前面加上window，`window`的属性或方法名也能顺利解析。

另外，你定义的所有全局变量都会放在window命名空间中，所以可以作为`window.myvariable`来引用（这里的myvariable就是你的全局变量名）。

再谈window.onload

这本书中目前为止经常用到一个`window.onload`事件处理程序。通过向`window.onload`属性指定一个函数，可以确保在页面加载和DOM完全建立之前不会运行代码。现在，`window.onload`语句中发生了很多事情，所以下面再来看一看这个语句，便于你理解：

这是我们的全局window对象。

onload是window对象的一个属性。

这是一个匿名函数，赋给onload属性。

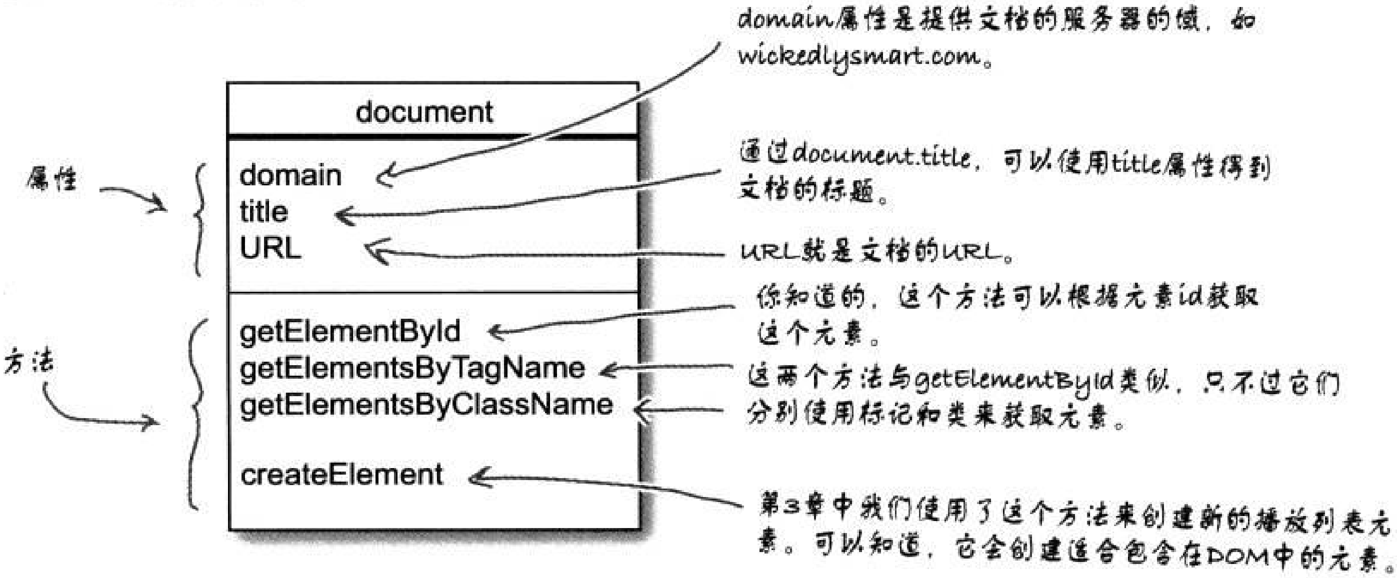
```
window.onload = function() {
    // code here
};
```

当然，一旦窗口完全加载页面并调用我们的匿名函数，就会执行函数体！

www.pin51.com 拼吾爱 最新最全的资源分享下载站

再谈document对象

document对象也是我们熟悉的面孔，这是用来访问DOM的对象。你已经看到，它实际上是window对象的一个属性，当然，我们没有用过window.document，因为并不需要这么做。下面来揭开面纱，简单地查看这个对象更有意思的一些属性和方法：



再谈document.getElementById

这一章最前面我们承诺过，读完这一章你一定会了解document.getElementById。嗯，你已经学习了函数、对象和方法，现在一切准备就绪！下面就来给出说明：

document是文档对象，这是一个内置的JavaScript对象，允许你访问DOM。

```
var div = document.getElementById("myDiv");
```

getElementById是一个方法……

……它取一个参数，一个<div>元素的id，并返回一个元素对象。



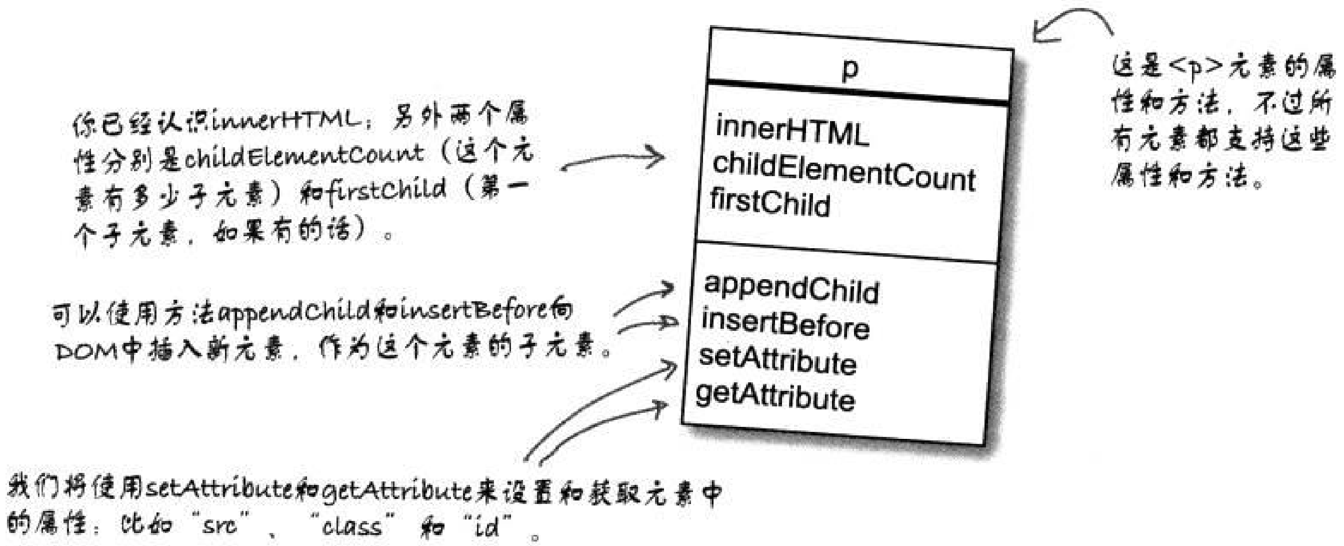
这个语法串看上去让人有些困惑，不过现在意思清楚多了，是不是？现在我们知道，div变量也是一个对象：一个元素对象。下面再来仔细查看这个对象。

pin5i.com 拼吾爱 最新资源分享下载站

再来考虑一个对象：元素对象

应该不会忘记，我们处理类似`getElementById`的方法时，它们返回的元素也是对象！嗯，也许你原先没有意识到这一点，不过既然现在已经知道了，可能会认为JavaScript中的一切都是对象，嗯，你想得非常正确。

你已经见过很多元素属性，如`innerHTML`属性。下面再来看另外一些重要的属性和方法：



there are no Dumb Questions

问：既然window是一个全局对象，这说明我可以使用它的属性和所有方法，根本不用先指定window，是这样吗？

答： 没错。是否在window对象的属性和方法前面加上window完全取决于你。对于类似alert的方法，所有人都知道它是什么，没有人会对这个方法加上window。另一方面，如果你在使用不那么有名的属性或方法，要想让你的代码更容易理解，就可以使用window。

问：这么说，理论上讲，我完全可以写成 onload = init，而不是 window.onload = init，对吗？

答： 是的。不过，对于这种特殊情况，我们并不推荐这么做，因为很多对象都有onload属性，所以如果在onload前面使用window的话，你的代码会更清楚。

问：我们之所以不说 window.onload = init()，原因是这会调用函数，而不是使用它的值，是这样吗？

答： 对的。如果函数名后面使用了小括号，比如init()，就是说你希望调用函数init。如果只是使用名而没有小括号，就会把这个函数值赋给onload属性。输入时差别很细微，但是它们的含义有天壤之别，所以一定要特别当心。

问：创建window.onload处理程序的两种方法中哪一种更好一些？是使用函数名还是匿名函数？

答： 并没有孰优孰劣，它们基本上都做同样的事情：把window.onload的值设置为页面加载时将运行的一个函数。如果出于某种原因以后要在程序中从另一个函数调用init，就需要定义一个init函数。否则，使用哪一种方法关系不大。

问： window和document之类的内置对象与我们自己创建的对象有什么区别？

答： 一个区别是内置对象遵循规范制定的原则，所以可以参考W3C规范来了解内置对象的所有属性和方法。另外，很多内置对象，比如String，可能有一些不可变的属性，不能修改。除此以外，所有对象都是对象。关于内置对象，很好的一点是这些对象已经为你创建好了。

没错，String是一个对象！可以查看一本好的JavaScript参考书来了解它的属性和方法的所有详细内容。



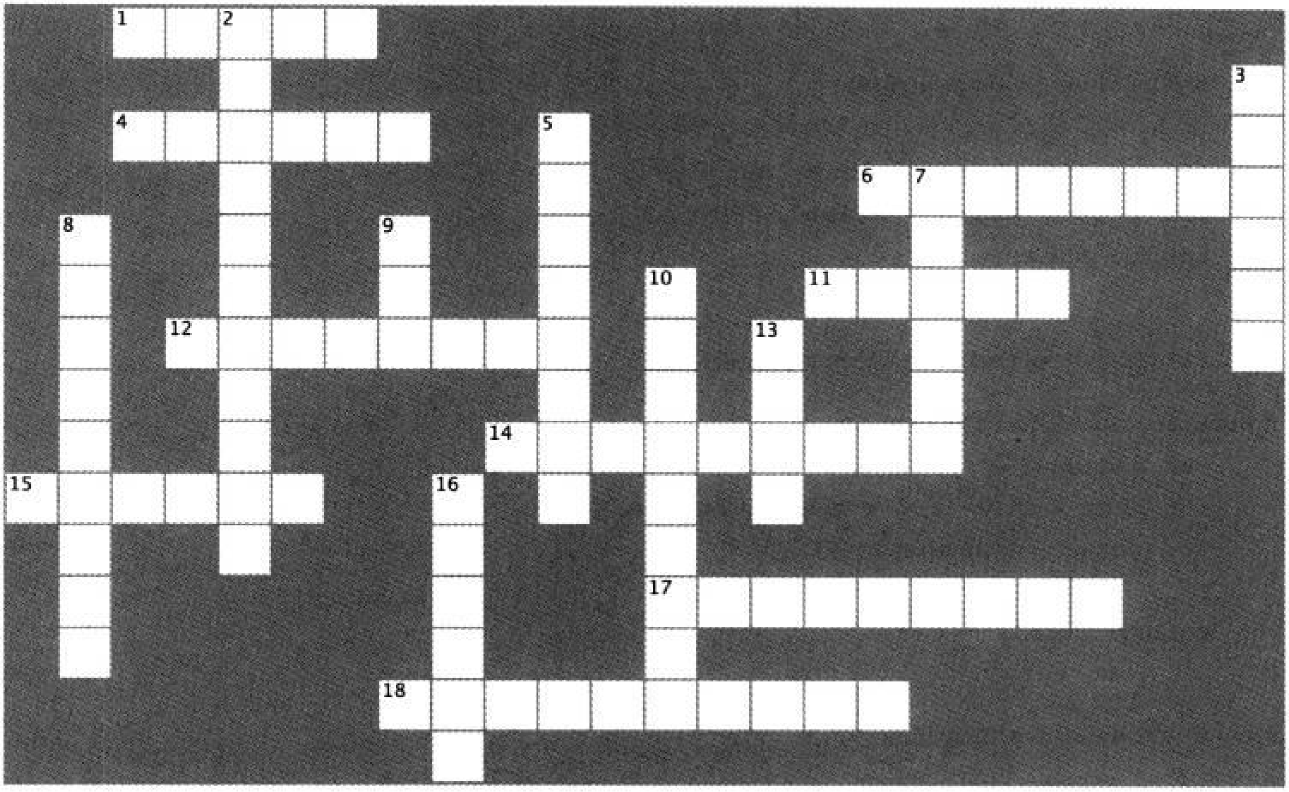
pin51.com 拼吾爱 最新最全的 下载站

BULLET POINTS

- 要创建一个函数，要使用function关键字，如果有参数，还要加上小括号来包含参数。
- 函数可以是命名函数，也可以是匿名的。
- 函数的命名规则与变量命名规则相同。
- 函数体放在大括号之间，包含完成函数工作的具体语句。
- 函数可以用return语句返回一个值。
- 要调用一个函数，需要使用函数名，并传入它需要的所有实参。
- JavaScript使用传值方式传递参数。
- 传递一个对象作为一个函数的实参时，比如dog，形参会得到这个对象引用的一个副本。
- 函数中定义的变量，包括形参，都称为局部变量。
- 函数外定义的变量称为全局变量。
- 局部变量在定义该变量的函数之外不可见。这称为变量的作用域。
- 如果声明一个局部变量时与一个全局变量同名，这个局部变量会遮蔽全局变量。
- 从页面链接到多个JavaScript文件时，所有全局变量都定义在同一个全局空间中。
- 如果赋一个新变量而没有使用var关键字，这个变量就是全局的，即使是在一个函数中首次赋值。
- 函数也是值，可以赋给变量、传递给其他函数、存储在数组中，还可以赋给对象属性。
- 对象是属性的集合。
- 可以使用点记法或[]记法访问对象的属性。
- 如果使用[ ]记法，要把属性名作为一个串用引号引起来，例如myObject["name"]。
- 可以改变一个属性的值、删除属性，或者向对象增加新属性。
- 可以使用一个for-in循环枚举对象的属性。
- 赋给一个对象属性的函数称为方法。
- 方法可以使用一个特殊的关键字this来引用调用这个方法的对象。
- 构造函数是创建对象的函数。
- 构造函数的任务是创建一个新对象，并初始化这个对象的属性。
- 要调用一个构造函数创建对象，需要使用new关键字。例如new Dog()。
- 这本书中一直在使用对象，包括document、window和很多元素对象。
- window对象是全局对象。
- document对象是window的一个属性。
- document.getElementById方法会返回一个元素对象。

HTML5填字游戏

真是激烈的一章，让我们充分了解了函数、对象、属性和方法，有太多东西需要记住。坐下来，放松一下，让你原先休息的那部分大脑运转起来。下面是第4章的填字游戏。



横向

1. 这些变量只在函数内部可用。
4. 真正的全局对象。
6. __________对象表示DOM。
11. 实参按_______________传递。
12. 使用这个关键字开始一个函数定义。
14. 没有return语句的函数返回___________。
15. 对象中的函数。
17. 没有名的函数。
18. 函数定义中要提供___________。

纵向

2. 这种函数会创建对象。
3. 函数可能包含也可能不包含这种语句。
5. 用点操作符把属性和函数调用串在一起。
7. window中的一个属性，我们会为它赋一个处理函数。
8. 函数调用中要提供__________。
9. _______ 操作符允许你访问一个对象的属性和方法。
10. 按约定，构造函数名首字母__________。
13. 对象方法中指示当前对象。
16. 变量作用域为到处都可见。

Sharpen your pencil Solution

你已经了解了函数，也知道了向行参传递实参，利用你现有的知识分析下面的代码。跟踪这个代码之后，在下面写出各个变量的值。下面是我们的答案。

```
function dogsAge(age) {
    return age * 7;
}

var myDogsAge = dogsAge(4);

function rectangleArea(width, height) {
    var area = width * height;
    return area;
}

var rectArea = rectangleArea(3, 4);

function addUp(numArray) {
    var total = 0;
    for (var i = 0; i < numArray.length; i++) {
        total += numArray[i];
    }
    return total;
}

var theTotal = addUp([1, 5, 3, 9]);

function getAvatar(points) {
    var avatar;
    if (points < 100) {
        avatar = "Mouse";
    } else if (points > 100 && points < 1000) {
        avatar = "Cat";
    } else {
        avatar = "Ape";
    }
    return avatar;
}
var myAvatar = getAvatar(335);
```

在这里写出各个变量的值……

myDogsAge = 28

rectArea = 12

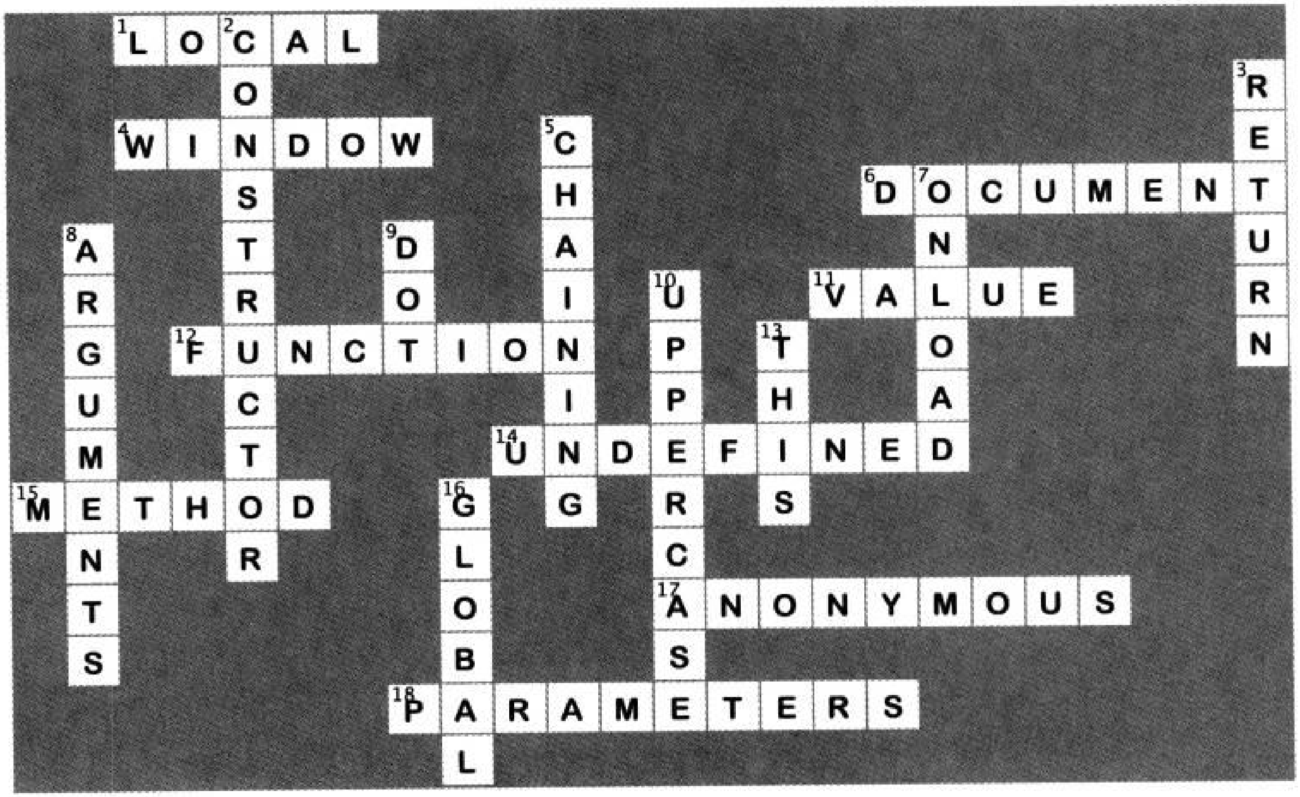
theTotal = 18

myAvatar = Cat

pin51.com 拼吾爱

拼吾爱

HTML5填字游戏答案



5 实现HTML位置感知

地理定位



无处可逃。有时，如果能知道你在什么位置，结果会大不相同（特别是对于一个Web应用）。这一章我们会告诉你如何创建位置感知的Web页面,有时你能精准地指出用户正站在哪个角落，有时只能确定他们位于哪个城区（不过总能知道在哪个城市）。当然了，有时你根本无法确定他们的位置，这可能是技术上的原因，也可能只是因为他们不希望你多管闲事。可以想想看。不管怎样，这一章我们开始研究一个JavaScript API：地理定位。带着你最好的位置感知设备（甚至可以是你的桌面PC），出发吧。

pin51.com 拼吾爱 最新最全的资源分享站

pin51.com 拼吾爱 最新最全资源分享下载站

你的用户正在移动，他们拿着位置感知的移动设备。最棒的应用应该能根据用户的位置提升用户体验。



位置，位置，位置

如果知道你的用户在哪里，这会大大改善Web体验：你可以向用户指明方向，对他们要去的地方给出建议，你会知道那里会不会下雨并建议室内活动，还可以让用户知道他们所在的区域内还有哪些人可能对某个活动感兴趣。使用位置信息的方式还有很多，没有做不到，只有想不到。

利用HTML5（以及基于JavaScript的地理定位API），可以很容易地在页面中访问位置信息。也就是说，正式开始之前，关于位置我们还需要了解几个问题。下面就来看一看……

there are no Dumb Questions

问： 我听说地理定位并不是一个真正的API，是吗？

答： 地理定位不算是现有HTML5标准的“直系”成员，不过不管怎样，它确实是W3C的一个标准，得到了广泛支持，而且几乎所有人都把地理定位当做一个重要的HTML5 API。所以它几乎就是一个真正的JavaScript API!

问： 地理定位API与Google Maps API一样吗？

答： 不一样。它们是完全不同的API。地理定位API只关注获取你的全球位置信息。Google Maps API是Google提供的一个JavaScript库，允许你访问所有Google Maps功能。所以，如果需要在一个地图中显示用户的位置，Google的API可以提供一种便捷方法来实现这个功能。

问： 我的设备会暴露我的位置，难道这不存在隐私问题吗？

答： 地理定位规范指出，所有浏览器必须得到用户的明确许可才能使用他们的位置。所以，如果你的代码使用了地理定位API，浏览器首先要做的就是确保用户同意共享他的位置。

问： 地理定位的支持程度如何？

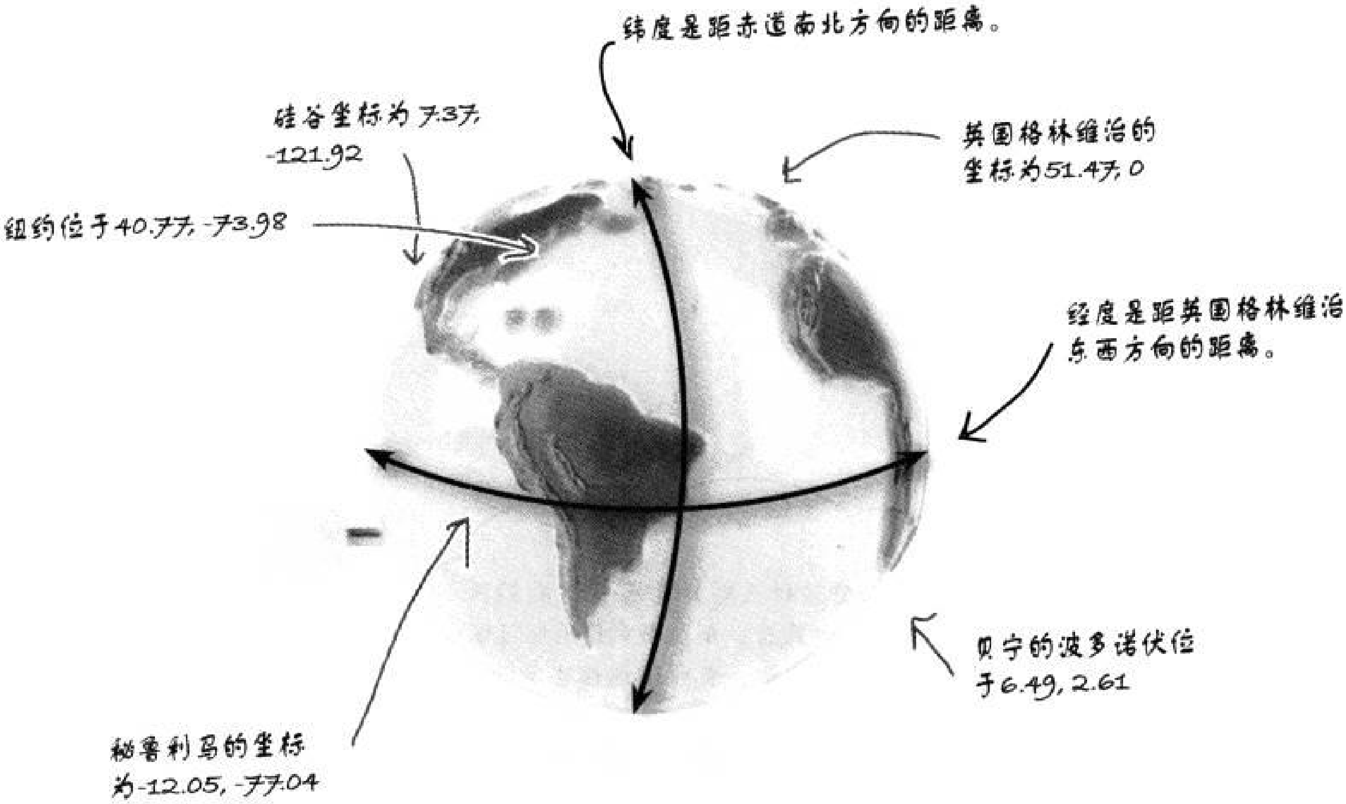
答： 地理定位得到了很好地支持。实际上，几乎每个现代浏览器都支持地理定位，这包括桌面和移动浏览器。你要确保使用浏览器的最新版本，如果确实如此，说明你很可能已经准备好了。

拼吾爱 最新最全的资源分享下载站 pin5i.com

纬度和经度……

要想知道你在哪里，需要有一个坐标系统，而且地球表面上要有这样一个坐标系统。很幸运，我们确实有这样一个东西，它使用纬度和经度共同构成坐标系统。纬度在南北方向指定地球上的一个点，经度则在东西方向上指定地球上的一个点。纬度从赤道开始测量，经度则以英国格林维治作为起点测量。地理定位API的工作就是使用这个经纬度提供任意时刻我们所在位置的坐标：

准确地讲，应该是格林维治皇家天文台。



纬度/经度特写

你可能见过用度/分/秒指定的纬度和经度，如(47°38'34",122°32'32")，还可能见过用小数值指定的经纬度，如(47.64, -122.54)。使用地理定位API时，都要使用小数值。如果需要将度/分/秒转换为小数，可以使用下面这个函数：

另外可以注意到，西经和南纬都用负数值表示。

```
function degreesToDecimal(degrees, minutes, seconds) {
  return degrees + (minutes / 60.0) + (seconds / 3600.0);
}
```

地理定位API如何确定你的位置

要做到位置感知并不要求你非得用最新的智能手机。即使桌面浏览器也一样能做到。你可能会问，桌面浏览器既没有GPS（全球定位系统），也没有其他方便的定位技术，怎么确定位置呢？嗯，所有浏览器（不论是设备浏览器，还是桌面浏览器）会使用不同的方法来确定你在哪里，只是有一些更精确一些，另外一些没那么精确而已。下面就来看一看：



GPS

很多较新的移动设备都支持全球定位系统（Global Positioning System），它能利用卫星提供极其精确的位置信息。位置数据可能包括高度、速度和朝向信息。不过，要使用GPS，你的设备必须能看到天空，而且可能需要花很长时间才能得到位置。另外GPS还很耗电。

IP地址

基于IP地址的位置信息使用一个外部数据库将IP地址映射到一个物理位置。这种方法的好处是任何地方都可以使用。不过，IP地址通常会解析为其他位置，比如你的ISP本地分局的位置，而不是你的具体位置。可以认为这种方法在城市级（甚至有时在街区级）很可靠。



拼吾爱

我的电话是老款的。没有装GPS。不过通过蜂窝基站三角定位，我的电话也能清楚地知道我在哪里，而且浏览器可以利用这个信息。



蜂窝电话

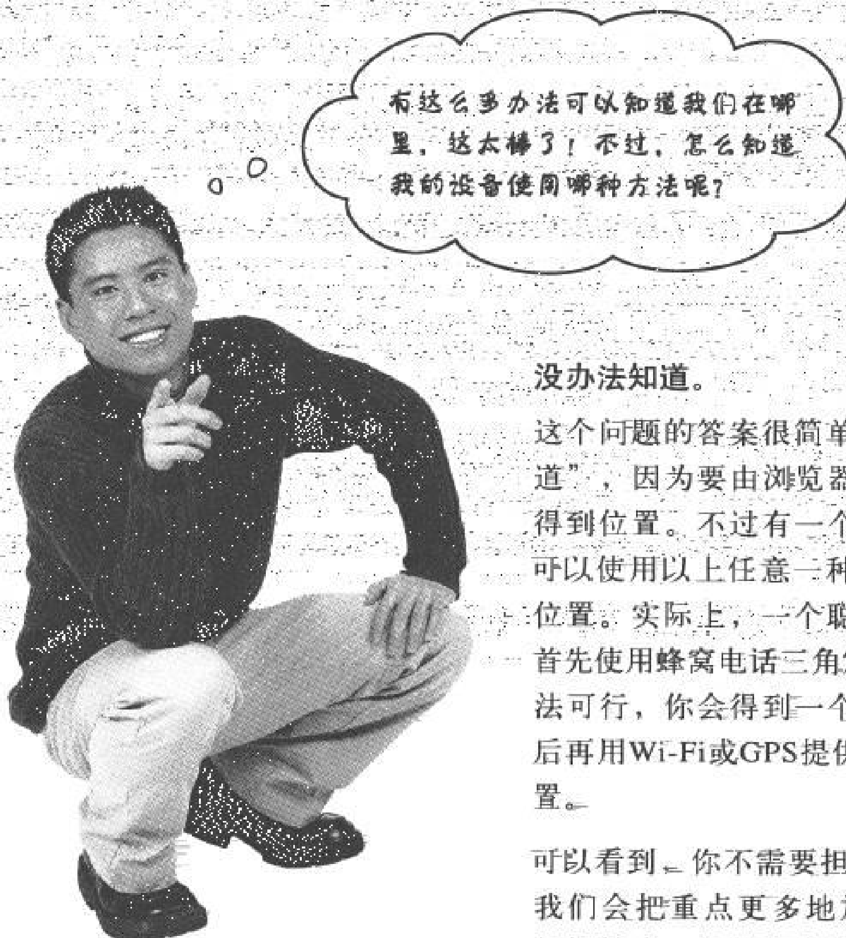
蜂窝电话三角定位可以根据你与一个或多个蜂窝电话基站的距离来确定你的位置（显然，基站越多，位置就越准确）。这种方法可能相当精确，在室内也能用（这与GPS不同），而且它比GPS也快得多。不过它也有一个缺点，如果你的位置很偏僻，附近只有一个蜂窝基站，位置的精度可能就不容乐观了。

我周转于不同的咖啡馆，带着我的笔记本电脑，通过无线订阅上网浏览。你通过三角定位所有这些无线运营商就能知道我在哪里。看起来很不错。



Wi-Fi

Wi-Fi定位使用一个或多个Wi-Fi接入点完成三角定位。这种方法可能很精确，在室内也可以用，而且速度很快。显然，这要求你相对处于静态（可能正在一家咖啡馆喝着大杯冰茶）。



没办法知道。

这个问题的答案很简单："你没办法知道"，因为要由浏览器实现来确定如何得到位置。不过有一个好消息，浏览器可以使用以上任意一种方法来确定你的位置。实际上，一个聪明的浏览器可能首先使用蜂窝电话三角定位，如果这种方法可行，你会得到一个大致的位置，然后再用Wi-Fi或GPS提供一个更精确的位置。

可以看到，你不需要担心位置如何确定，我们会把重点更多地放在位置精度上。根据精度，你能确定这个位置对你来说有什么用。别走开，稍后就会回来讨论精度。

www.pin5i.com 拼吾爱 最新编程资源分享下载站

Sharpen your pencil

请考虑你现有的HTML页面和应用（或者你想要创建的应用），
如何在其中结合位置信息？

- ☐ 允许用户与附近的其他人共享这个位置信息。
- ☐ 让用户更容易地找到本地资源或服务。
- ☐ 跟踪用户在哪里活动。
- ☐ 告诉用户他们来的方向。
- ☐ 使用位置确定用户的其他人口统计特征。
- ☐ ....
- ☐ ....
- ☐ ....
- ☐ ....

在这里写出你的想法！

pin51.com

你到底在哪里？

嗯，你当然知道自己在哪里，不过下面来看你的浏览器认为你在哪里。为此，只需要创建一个很小的HTML：



最前面还是老一套，包括指向一个文件（myLoc.js）的链接，我们的所有JavaScript都放在这个文件中，另外还有一个样式表myLoc.css，它能让页面看起来更漂亮。

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Where am I?</title>
  <script src="myLoc.js"></script>
  <link rel="stylesheet" href="myLoc.css">
</head>
<body>
  <div id="location">
    Your location will go here.
  </div>
</body>
</html>
```

我们将把地理定位代码写在myLoc.js中。

使用这个<div>输出你的位置。

将所有这些HTML代码放在一个名为myLoc.html的文件中。

现在来创建myLoc.js，并写一些代码。我们先很快写完代码，然后再回过头来仔细研究。把下面的代码增加到你的myLoc.js文件中：

一旦浏览器加载页面，我们将调用函数getMyLocation。

```
window.onload =  getMyLocation;

function getMyLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(displayLocation);
    } else {
        alert("Oops, no geolocation support");
    }
}
```

利用这个检查来确保浏览器支持地理定位API。如果navigator.geolocation对象存在，说明浏览器支持这个API。

如果浏览器支持地理定位API，则调用getCurrentPosition方法，并传入一个处理函数displayLocation。稍后就会实现这个函数。

displayLocation函数就是将要操纵位置的处理程序。

如果浏览器不支持地理定位，就向用户弹出一个提醒。

这就是我们的处理程序，浏览器得到一个位置时就会调用这个函数。

向getCurrentPosition的处理程序传入一个位置，其中包含位置的纬度和经度（以及一些精度信息，后面将会介绍）。

```
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;

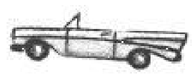
    var div = document.getElementById("location");
    div.innerHTML = "You are at Latitude: " + latitude + ", Longitude: " + longitude;
}
```

从position.coords对象得到位置的纬度和经度。

然后从HTML获得<div>……

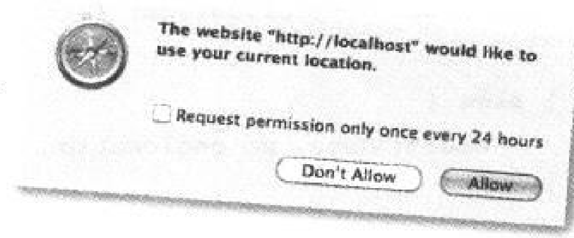
……现在我们只是使用innerHTML将location <div>的内容设置为你的位置。

试一试你的位置



输入这个代码，试一试这个新的位置感知页面。

第一次运行一个地理定位Web应用时，你会注意到浏览器会请求得到你的许可来使用你的位置。这是一个浏览器安全检查，你完全可以告诉浏览器不允许使用你的位置信息。不过，假设你想测试这个Web应用，应该会点击Allow或Yes。确定后，这个应用会显示出你的位置，如下所示：



根据你使用的浏览器，这个许可请求可能有所不同，不过基本上是类似的。



这就是你的位置！显然，你的位置可能与我们的不同（如果不是这样，反倒会让我们担心）。

如果没有得到你的位置，而且假设你已经反复检查了输入错误之类的问题，别着急，再过几页我们会给出一些代码来调试这个问题……

要记住，并不总是立即就能得到位置，可能需要等一会儿……

pin51.com 拼吾爱



我们刚才做了……

既然有了一些地理定位代码，而且已经成功地运行（再说一次，如果你还没有看到位置，先别着急，稍后就会介绍一些调试技术），下面更详细地分析这个代码：

1 **如果要写地理定位代码，首先需要知道的是“这个浏览器支持地理定位吗？”为此，我们利用了这样一个事实：只有当浏览器支持地理定位时，它的navigation对象中才有一个geolocation属性。**

所以我们可以查看geolocation属性是否存在，如果存在，就使用这个属性；否则，要让用户知道：

```
if (navigator.geolocation) {
    ......
} else {
    alert("Oops, no geolocation support");
}
```

可以使用一个小测试来查看是否支持地理定位（如果不支持，navigator.geolocation会计算为null，这个条件将失败）。

如果有这个属性，可以直接利用，如果没有，则要让用户知道。

2 **现在，如果确实有一个navigator.geolocation属性，可以继续使用。实际上，navigator.geolocation属性是一个对象，其中包含整个地理定位API。这个API支持的主要方法是getCurrentPosition，它的工作是获取浏览器位置。下面更深入地了解这个方法，它有3个参数，其中两个是可选的：**

要记住，API就是带属性和方法的对象。现在你是不是很庆幸已经提前完成了JavaScript培训！

successHandler是一个函数，如果浏览器能成功地确定你的位置就会调用这个函数。

errorHandler是另外一个函数，如果出了问题，浏览器无法确定你的位置，则会调用这个函数。

```
getCurrentPosition(successHandler, errorHandler, options)
```

这两个参数是可选的，正因如此，之前我们不需要这两个参数。

options参数允许你定制地理定位方法。

www.pin51.com 拼吾爱 最新最全资源分享下载站

3 下面再来看我们的getCurrentPosition方法调用。对现在来说，我们只提供了successHandler参数来处理成功得到浏览器位置的情况。稍后还会分析浏览器无法找到位置的情况。

```
if (navigator.geolocation) {

    navigator.geolocation.getCurrentPosition(displayLocation);

}
```

还记得第4章介绍过的串链吗？我们使用navigator对象来访问geolocation对象，这只是navigator的一个属性。

这里在调用geolocation对象的getCurrentPosition方法，并且提供了一个实参，即成功回调。

geolocation确定了你的位置时，会调用displayLocation。

注意到了吗？这里把一个函数传递到另一个函数。应该记得第4章说过，函数也是值，所以完全可以这样做，没有任何问题。

4 现在来看成功处理程序displayLocation。调用displayLocation时，地理定位API向它传入一个position对象，其中包含有关浏览器位置的信息，包括一个有纬度和经度的coordinates对象（它还包括其他一些值，后面再做讨论）。

```
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;

    var div = document.getElementById("location");
    div.innerHTML = "You are at Latitude: " + latitude + ", Longitude: " + longitude;
}
```

position是地理定位API传入成功处理程序的对象。

position对象有一个coords属性，其中包含指向coordinates对象的一个引用……

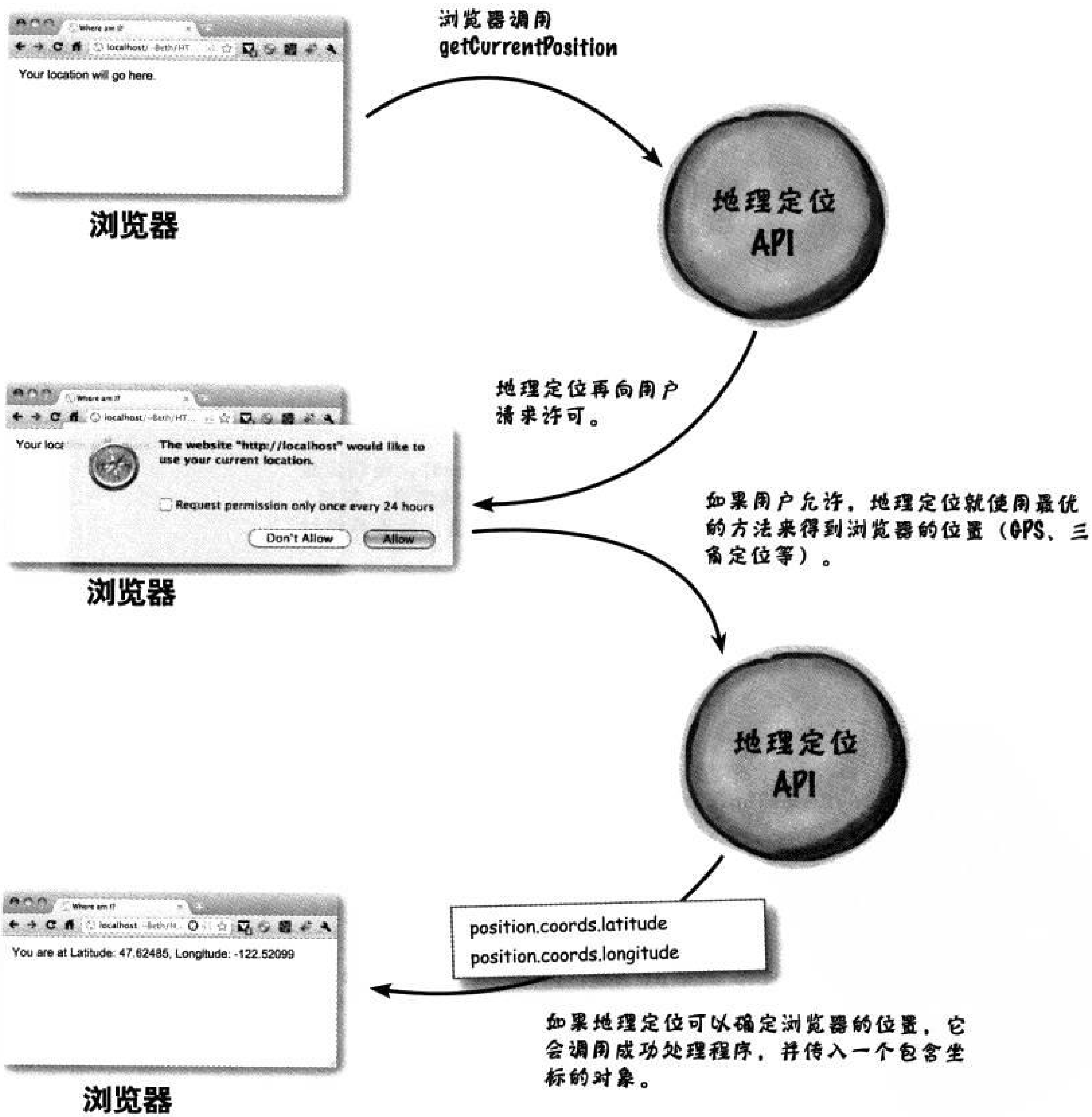
……coordinates对象包含了你的纬度和经度。

这一部分我们相信对你现在来说已经是小菜一碟了。这里只是得到坐标信息，并在页面的一个<div>中显示。

pin51.com 拼吾爱

如何集成

既然已经完成了代码，下面来看运行时是如何工作的：



试一试诊断



用到地理定位时，并不是每一个测试都能成功，即使你的第一个测试是成功的，接下来也有可能在某个地方出错。我们想帮帮你，所以为你创建了一个很小的诊断测试，可以把它直接加到你的代码中。这样一来，如果你遇到麻烦，就可以利用这个诊断代码找出答案。即使你自己没有麻烦，但你的某个用户可能会遇到问题，你肯定想知道如何在代码中解决这个问题。所以增加下面的代码，如果遇到麻烦，一旦诊断到问题请填写诊断表。

要创建这个诊断测试，我们需要为getCurrentPosition方法调用增加一个错误处理程序。只要地理定位API在确定你的位置时遇到问题就会调用这个处理程序。如下增加这个错误处理程序：

为getCurrentPosition调用增加第二个参数，名为displayError。这是地理定位未能找到位置时要调用的一个函数。

```
navigator.geolocation.getCurrentPosition(displayLocation, displayError);
```

现在要编写这个错误处理程序。为此，需要知道的是，geolocation将会向你的处理程序传入一个error对象，其中包含一个数值码，描述了它未能确定浏览器位置的原因。根据这个数值码，还可以提供一个消息，包含有关这个错误的更多信息。可以如下在处理程序中使用这个error对象：

这是我们的新处理程序，地理定位API会向它传入一个错误。

```
function displayError(error) {
    var errorTypes = {
        0: "Unknown error",
        1: "Permission denied by user",
        2: "Position is not available",
        3: "Request timed out"
    };
    var errorMessage = errorTypes[error.code];
    if (error.code == 0 || error.code == 2) {
       errorMessage = errorMessage + " " + error.message;
    }
    var div = document.getElementById("location");
    div.innerHTML = errorMessage;
}
```

error对象有一个code属性，其中包含一个0~3的数。这是用JavaScript为各个错误码关联一个错误消息的好办法。

我们创建了一个对象，包括4个属性，分别名为0~3。这些属性是串，包含了我们希望与各个错误码关联的错误消息。

使用error.code属性，将一个错误消息串赋给一个新变量errorMessage。

对于错误0和2，有时error.message属性中会有一些额外的信息，所以把这些信息增加到errorMessage串。

然后把这个消息增加到页面，让用户知道。

pin51.com 拼吾爱

运行这个测试之前，再仔细研究可以得到哪些类型的错误。

```
var errorTypes = {
    0: "Unknown error",
    1: "Permission denied by user",
    2: "Position is not available",
    3: "Request timed out"
};
```

这是“全包型”错误，如果其他错误都不合适就会使用这个错误。查看error.message属性来了解更多信息。

这表示用户拒绝了使用位置信息的请求。

这说明浏览器尝试过，但是没能得到你的位置。同样的，查看error.message来得到更多信息。

最后，地理定位有一个内部超时设置，如果超出了这个时限但还没能确定位置，就会导致这个这个错误。

这一章后面就会看到如何改变地理定位的默认超时设置。

输入前面的诊断测试之后，可以试一试。显然，如果接收到一个位置，说明一切正常，你不会看到任何错误。也可以拒绝浏览器使用你的位置的请求，来强制产生一个错误。或者可以更有创意一点，比方说，拿着你的GPS电话进入室内同时关闭网络。在最坏的情况下，如果你等待了很长时间而没有得到任何位置或错误消息，很有可能是因为超时值过长，你一直在等待达到这个超时时间。本章后面会介绍如何缩短这个超时时限。



在这里填写你的诊断结果

- ☐ 我不允许使用我的位置。
- ☐ 不能得到我的位置。
- ☐ 过几秒后，我收到一个消息，指出请求超时。
- ☐ 什么都没有发生，没有位置，也没有错误提示。
- ☐ 其他 ______________________

www.nn51.com 拼吾爱 最新最全电子书免费下载

Watch it!

要在一个移动设备上测试你的地理定位代码，需要有一个服务器。

除非有办法把你的HTML、JavaScript和CSS文件直接加载到你的移动设备上，否则要想测试这些代码，最容易的方法就是把它们放在一个服务器上（如果愿意，你可以先简单看看下一章，了解如何建立你自己的服务器），并从服务器访问代码。如果你已经有一个服务器，而且希望这样做，建议你现在就着手完成。另一方面，如果你无法找到其他服务器，我们可以保证在Wickedly Smart服务器上肯定能得到这个代码，这样就能在你的移动设备上测试了。也就是说，建议你先在桌面浏览器中运行代码，一旦成功运行，再使用服务器（可以是你自己的服务器，也可以是Wickedly Smart服务器）在你的移动设备上进行测试。

作为第一个测试（包括错误诊断），将设备指向http://wickedlysmart.com/hfhtml5/chapter5/latlong/myLoc.html。

there are no Dumb Questions

问：这个应用返回了我的位置，但是纬度和经度不太对，为什么？

答： 你的设备和位置服务提供商会采用多种方法计算你的位置，有些会精确一些，有些不那么精确。GPS通常是最精确的。我们会介绍一种确定精度的方法，位置服务会返回估计的这个精度，作为位置对象的一部分，使你能了解位置数据可能的精确程度。

找出我们的秘密位置……

既然已经有了基础，下面对位置做一些更有意思的处理。看看你与我们在Wickedly Smart总部的秘密位置相距多远，怎么样？为此，我们需要总部坐标，而且需要知道如何计算两个坐标之间的距离。首先，再在HTML中增加一个<div>：



Wickedly Smart总部位于47.62485, -122.52099。

```
   ⋮
<body>
  <div id="location">
    Your location will go here.
  </div>
  <div id="distance">
   Distance from WickedlySmart HQ will go here.
  </div>
</body>
</html>
```

把这个新<div>增加到你的HTML。

pin51.com 拼吾爱



一些成品代码：计算距离

想知道如何计算地球上两点间的距离吗？你会发现有关的细节很有意思，不过这有些超出这一章的范畴。所以，我们会给你一些成品代码来完成这个计算。要计算两个坐标之间的距离，几乎所有人都会使用半正矢（Haversine）公式，下面实现了这个公式。如果需要知道两个坐标之间的距离，就可以使用这个代码来计算：

这个函数取两个坐标，一个起点坐标和一个终点坐标，并返回二者之间的距离（单位为千米）。

```
function computeDistance(startCoords, destCoords) {
    var startLatRads = degreesToRadians(startCoords.latitude);
    var startLongRads = degreesToRadians(startCoords.longitude);
    var destLatRads = degreesToRadians(destCoords.latitude);
    var destLongRads = degreesToRadians(destCoords.longitude);

    var Radius = 6371; // radius of the Earth in km
    var distance = Math.acos(Math.sin(startLatRads) * Math.sin(destLatRads) +
                   Math.cos(startLatRads) * Math.cos(destLatRads) *
                   Math.cos(startLongRads - destLongRads)) * Radius;

    return distance;
}
```

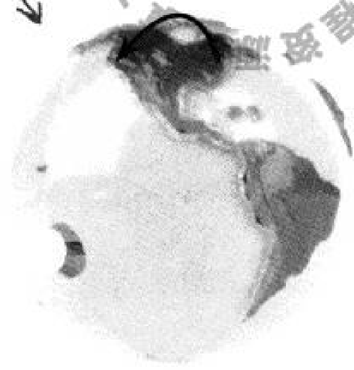
```
function degreesToRadians(degrees) {
    var radians = (degrees * Math.PI)/180;
    return radians;
}
```

在“画布”一章中还会看更多地使用了这个函数。

把这个代码增加到你的myLoc.js文件。

编写代码查找距离

我们想计算你与我们之间的距离，如箭头所指。



既然有了一个函数来计算两个坐标之间的距离，下面来定义我们在WickedlySmart总部的位置（也就是本书作者所在位置），也可以输入以下代码：

```
var ourCoords = {
        latitude: 47.624851,
        longitude: -122.52099
};
```

这里为Wickedly Smart总部的位置坐标定义了一个字面量对象。把它作为一个全局变量增加到myLoc.js文件的最前面。

现在来编写代码，我们所要做的就是把你的位置和我们的位置坐标传递到computeDistance函数：

```
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;

    var div = document.getElementById("location");
    div.innerHTML = "You are at Latitude: " + latitude + ", Longitude: " + longitude;

    var km = computeDistance(position.coords, ourCoords);
    var distance = document.getElementById("distance");
    distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ";
}
```

这里将你的位置坐标以及我们的坐标传递到computeDistance。

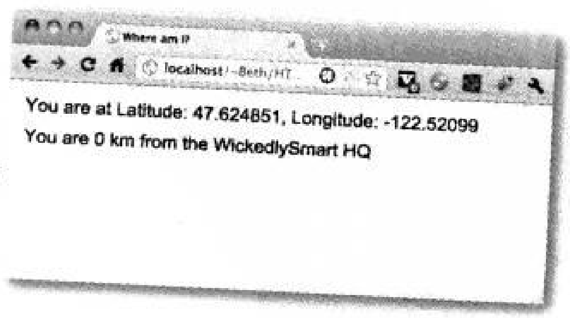
然后得到结果，并更新distance <div>的内容。

试一试位置代码



现在来试一试这个新代码。把这个代码增加到myLoc.js，然后在浏览器中重新加载myLoc.html。你会看到你的位置，以及你与我们的距离。

你的位置和距离显然会不同，这取决于你具体在哪里。



在线测试：http://wickedlysmart.com/hfhtml5/chapter5/distance/myLoc.html

pin51.com 拼吾爱



提供位置地图

前面我们说过，地理定位API相当简单，它提供了一种方法来查找你在哪里（你会看到，它还能够跟踪你的位置），不过这个API并没有提供任何工具为你的位置给出可视化表示。为此，我们需要依赖一个第三方工具，可以猜到，目前为止在这方面Google Maps是最流行的工具。显然，Google Maps不是HTML5规范的一部分，不过它确实能与HTML5很好地互操作，所以我们不介意时常会临时转向，展示如何将它与地理定位API集成。如果你想换换口味，了解这些内容，首先可以把这个API增加到HTML文档的首部，接下来我们会讨论如何向你的页面增加一个地图：

```
<script src="http://maps.google.com/maps/api/js?sensor=true"></script>
```

这是Google Maps JavaScript API的位置。

一定要照原样输入，包括sensor查询参数（如果没有它，这个API将无法正常工作）。我们会使用sensor=true，因为代码中要用到你的位置。如果只是使用地图而不需要你的位置，可以输入sensor=false。

脱离跑道 临时转向

如何向页面增加地图

既然已经链接到Google Map API，就可以通过JavaScript得到Google Maps的所有功能。不过，我们需要一个地方容纳Google Map，为此要定义一个元素来放置地图。

```
   ⋮
<body>
  <div id="location">
    Your location will go here.
  </div>
  <div id="distance">
   Distance from WickedlySmart HQ will go here.
  </div>
  <div id="map">
  </div>
</body>
</html>
```

这里是一个<div>。需要说明，我们已经在myLoc.css中定义了一些样式，将map <div> 设置为400px×400px，而且有一个黑色边框。

准备创建一个地图……

要创建地图，我们需要两样东西：纬度和经度（我们已经知道如何得到这些信息），另外需要一组选项来描述希望如何创建地图。下面先看纬度和经度。我们已经知道如何用地理定位API得到纬度和经度，不过Google API希望纬度和经度包装在单独的对象中。要创建这样一个对象，可以使用Google提供的一个构造函数：

```
var googleLatAndLong = new google.maps.LatLng(latitude, longitude);
```

不要忘记，构造函数首字母要大写。

Google Maps API的所有方法前面都有google.maps。

这里是构造函数，它取我们的纬度和经度，并返回一个新对象，其中同时包含纬度和经度。

Google提供了一些选项，可以设置这些选项来控制如何创建地图。例如，可以控制初始地图视图的放大或缩小程度、地图在哪里居中、地图的类型（如道路地图、卫星视图，或者二者兼有）。可以创建这些选项如下：

```
var mapOptions = {
    zoom: 10,
    center: googleLatAndLong,
    mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

zoom选项可以指定为0~21的一个值。可以用不同的zoom尝试一下：较大的数对应进一步放大（能看到更多细节）。10大约对应"城市"规模。

这里是我们刚创建的新对象。我们希望地图在这个位置居中。

还可以试试SATELLITE和HYBRID作为这个选项值。

脱离跑道
临时转向

显示地图

下面把所有这些代码放在一个新函数showMap中，这个函数取一组坐标，在你的页面上显示一个地图：

```
var map;

function showMap(coords) {
    var googleLatAndLong =
            new google.maps.LatLng(coords.latitude,
                                   coords.longitude);
    var mapOptions = {
        zoom: 10,
        center: googleLatAndLong,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var mapDiv = document.getElementById("map");
    map = new google.maps.Map(mapDiv, mapOptions);
}
```

我们声明了一个全局变量map，它会包含我们将创建的Google地图。稍后你会看到如何使用这个地图。

使用coords对象的latitude和longitude……

……用它们来创建一个google.maps.LatLng对象。

用我们希望的地图选项创建mapOptions对象。

最后，从DOM获取map <div>，把它和mapOptions传递到Map构造函数，创建google.maps.Map对象。这会在我们页面上显示地图。

将这个新Map对象赋给全局变量map。

这是Google的API提供的另一个构造函数，它取一个元素和我们的选项，创建并返回一个地图对象。

把这个代码增加到JavaScript文件的最后面。现在我们只需要把它与现有的代码联系起来。下面通过编辑displayLocation函数来做到：

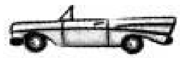
```
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;

    var div = document.getElementById("location");
    div.innerHTML = "You are at Latitude: " + latitude + ", Longitude: " + longitude;

    var km = computeDistance(position.coords, ourCoords);
    var div = document.getElementById("distance");
    distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ";

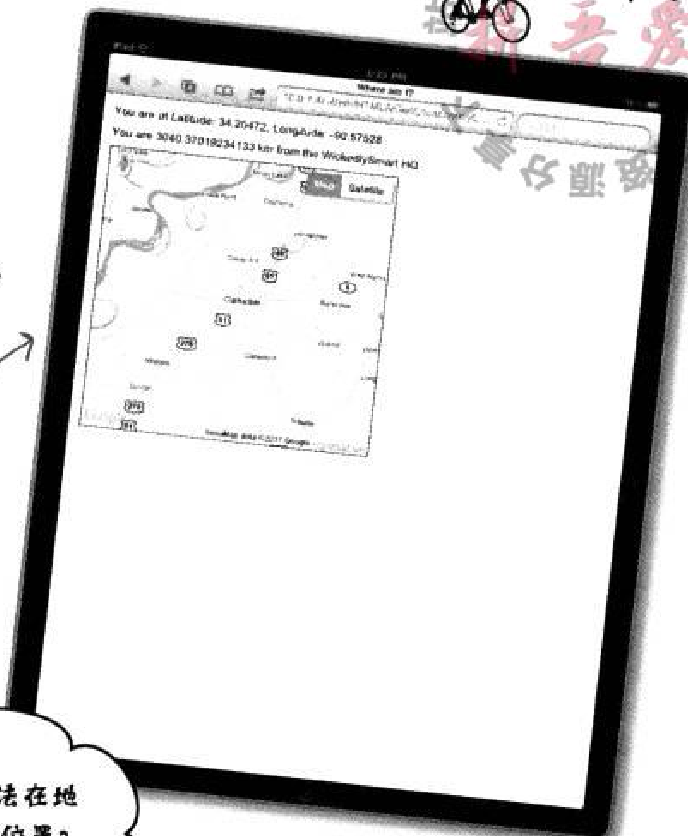
    showMap(position.coords);
}
```

更新页面上的其他<div>之后，从displayLocation调用showMap。

试一试这个新显示 

确保已经增加了上一页的所有代码，另外向HTML增加了新的map <div>；然后重新加载你的页面，如果浏览器可以确定你的位置，你就会看到一个地图。

这是我们的新Google Map!



我们显示了车手的位置在34.20472, -90.57528。当然你也可能在其他地方。

真不错！有没有办法在地图上看到我的准确位置？就像有那种大头钉一样？



你真的希望这个东西靠近你的自行车吗？

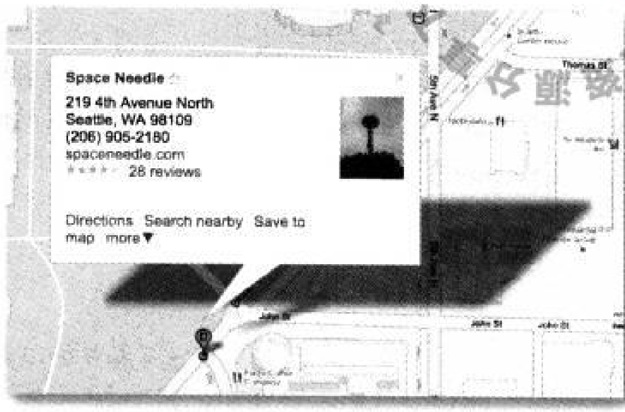


在线测试：http://wickedlysmart.com/hfhtml5/chapter5/map/myLoc.html

脱离跑道 临时转向

加一个大头钉……

如果可以准确地看到你在地图上的位置，就会更有用。如果你用过Google Maps，可能对它使用大头钉标记搜索项位置的做法很熟悉。例如，如果你搜索华盛顿西雅图的太空针高塔（Space Needle），就会得到一个地图，其中在这个城市靠近太空针高塔的地区有一个大头钉，点击这个大头钉，会看到一个信息窗口，其中会显示有关这个搜索项的更多详细信息。嗯，大头钉也叫做标记（marker），这是Google Maps API提供的众多特性之一。



在Google Maps中搜索一项时，你会看到一个红色的大头钉标记搜索结果所在的位置。

要增加一个带弹出信息窗口的标记，需要编写一点代码，因为你必须创建这个标记和信息窗口，还要增加标记点击事件的处理程序（它会打开这个信息窗口）。因为我们只是临时转向，所以只会简要介绍这个内容，不过，既然已经读到这里，你应该已经掌握学习这个内容所需的全部知识了！

① 首先创建一个新函数addMarker，然后使用Google API创建一个标记：

addMarker函数取一个地图、一个google样式的纬度和经度，标记的标题，以及信息窗口的一些内容。

```
function addMarker(map, latlong, title, content) {
    var markerOptions = {
        position: latlong,
        map: map,
        title: title,
        clickable: true
    };
    var marker = new google.maps.Marker(markerOptions);
}
```

用纬度和经度、地图、标题以及是否希望这个标记可点击来创建一个options对象……

……这里把它设置为true，因为我们希望点击这个大头钉时能够显示一个信息窗口。

然后使用Google API提供的另一个构造函数创建marker对象，并传入我们创建的markerOptions对象。

www.jb51.com 最新最全的资源分享 脚本之家

② 接下来，定义特定于信息窗口的一些选项，来创建这个信息窗口，然后用Google API创建一个新的InfoWindow对象。将下面的代码增加到你的addMarker函数：

```
function addMarker(map, latlong, title, content) {
  :
  :
    var infoWindowOptions = {
        content: content,
        position: latlong
    };

    var infoWindow = new google.maps.InfoWindow(infoWindowOptions);

    google.maps.event.addListener(marker, "click", function() {
        infoWindow.open(map);
    });
}
```

其他代码仍然放在这里，我们想节省些篇幅（少用些纸，相应地也能少砍些树……

现在要为信息窗口定义一些选项。

我们需要内容……

……还有纬度和经度。

用它来创建信息窗口。

接下来使用Google Maps addListener方法为点击事件增加一个“监听者”。监听者就像一个处理程序，类似前面已经见过的onload和onclick。

向监听者传入一个函数，用户点击标记时就会调用这个函数。

点击标记时，会调用这个函数，在地图上打开信息窗口。

③ 现在要做的就是从showMap调用addMarker函数，要确保为这4个行参传入了正确的实参。将这个代码增加到showMap函数的最下面：

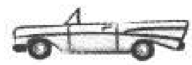
```
var title = "Your Location";
var content = "You are here: " + coords.latitude + ", " + coords.longitude;
addMarker(map, googleLatAndLong, title, content);
```

传入使用Google maps API创建的map和googleLatAndLong对象……

……和一个标题串，以及标记的一个内容串。

脱离跑道
临时转向

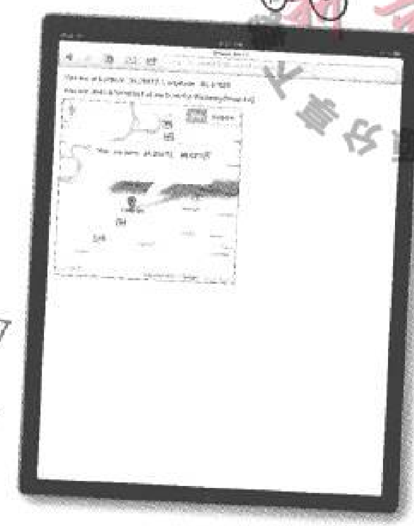
测试标记



增加addMarker的所有代码之后，更新showMap来调用addMarker，并重新加载页面。你会看到一个地图，而且你的位置上有一个标记。

可以试着点击这个标记，会得到一个弹出窗口，给出你的纬度和经度。

这很棒，因为现在你能准确地知道你在哪里（万一你迷路了或者出了什么状况……）



这就是我们的地图，包含标记和弹出的信息窗口。

在线测试：http://wickedlysmart.com/hfhtml5/chapter5/marker/myLoc.html

用Google Maps API还能做另外一些很酷的事情



使用Google Maps API可以做很多事情，我们只是稍稍触及皮毛，尽管这个API超出了本书的范畴，不过，如果你能自己深入研究会很有好处。下面简要介绍一些工作，你可以考虑用这个API完成这些工作，这里还指出了可以从哪里开始。

控件：默认地，你的Google地图会包括很多控件，比如缩放控件、平移控件、切换地图和卫星视图的控件，甚至还会有街景视图控件（缩放控件上面的小人图标）。可以从JavaScript通过程序访问这些控件，在你的应用中使用。

服务：在Google Maps中查找过方向吗？如果有过，说明你已经用过方向（Directions）服务。你可以访问方向以及其他服务，如通过Google Maps服务API访问距离和街区视图。

覆盖：覆盖会在Google地图的上面提供另一个视图，比方说，一个热图覆盖。如果你正在路上，可以用交通覆盖来检查交通阻塞情况。使用Google Maps覆盖API可以创建定制覆盖，如定制标记、你的照片，以及你能想到的任何其他东西。

所有这些都可以通过Google Maps JavaScript API得到。要进一步试验，可以查看以下地址提供的文档：
http://code.google.com/apis/maps/documentation/javascript/

nn51.com 拼吾爱



地理定位闪亮登场

本周访谈：
与一个想有所作为的HTML5 API的对话

Head First: 欢迎你，地理定位。我得先说一句，能在这里看到你我真的有点奇怪。

地理定位: 为什么呢？

Head First: 你甚至不能“正式”算是HTML5规范的一部分，但你居然在这里出现，要知道我们第一次专门用一章来介绍一个API，而你竟然就是我们第一个要介绍的主角！怎么会这样？

地理定位: 嗯，你说得对，我确实是在另一个规范里定义，而不是HTML5规范，不过我真的是W3C的官方规范。另外，你可以四处看看，所有那些数得着的移动设备都已经在浏览器中实现了我。我是说，如果一个移动Web应用没有我还能有什么意义呢？

Head First: 那么哪些Web应用会用到你呢？

地理定位: 实际上，人们在移动中使用的大部分应用都会用到我；从允许你更新状态并包括地理信息的应用，到记录照片在哪里拍摄的相机应用，再到查找当地朋友或者允许你在不同位置“登录”的社交应用，你都能找到我的身影。嘿，人们甚至用我来记录他们在哪里骑车、跑步、用餐或者了解他们要去哪里。

Head First: 你的API看起来有点简单，我的意思是，你总共就只有这么几个方法和属性吗？

地理定位: 浓缩才是精华。你见过有人对我抱怨吗？没有吧！ 我花了很久来了解每个开发人员需要什么，也知道如何建立位置感知应用。另外，小就意味着速度快，而且容易学，不是吗？为什么把我选做第一个单独用一章来介绍的API，可能就是因为这个原因吧？

Head First: 我们来谈谈支持情况。

地理定位: 这个话题没什么多说的，因为几乎每个浏览器都支持我，不论是桌面浏览器还是移动浏览器。

Head First: 好吧，有一件事我一直想问你：如果一个设备上没有GPS，要你有什么用呢？

地理定位: 这里存在一个严重的误解，以为我多少会依赖GPS。如今通过蜂窝电话三角定位、使用IP地址等可以有很多很棒的方法来确定位置。如果你有GPS，那当然好，实际上这样我就能提供更大帮助。不过，即使没有GPS，还是有很多办法来得到位置的。

Head First: 更大帮助？

地理定位: 如果你有一个足够好的移动设备，我可以告诉你纬度、方向、速度，诸如此类的信息我都能提供。

Head First: 假设所有这些方法都不奏效，也就是说，GPS、IP地址、三角定位都不能用，你还能做什么呢？

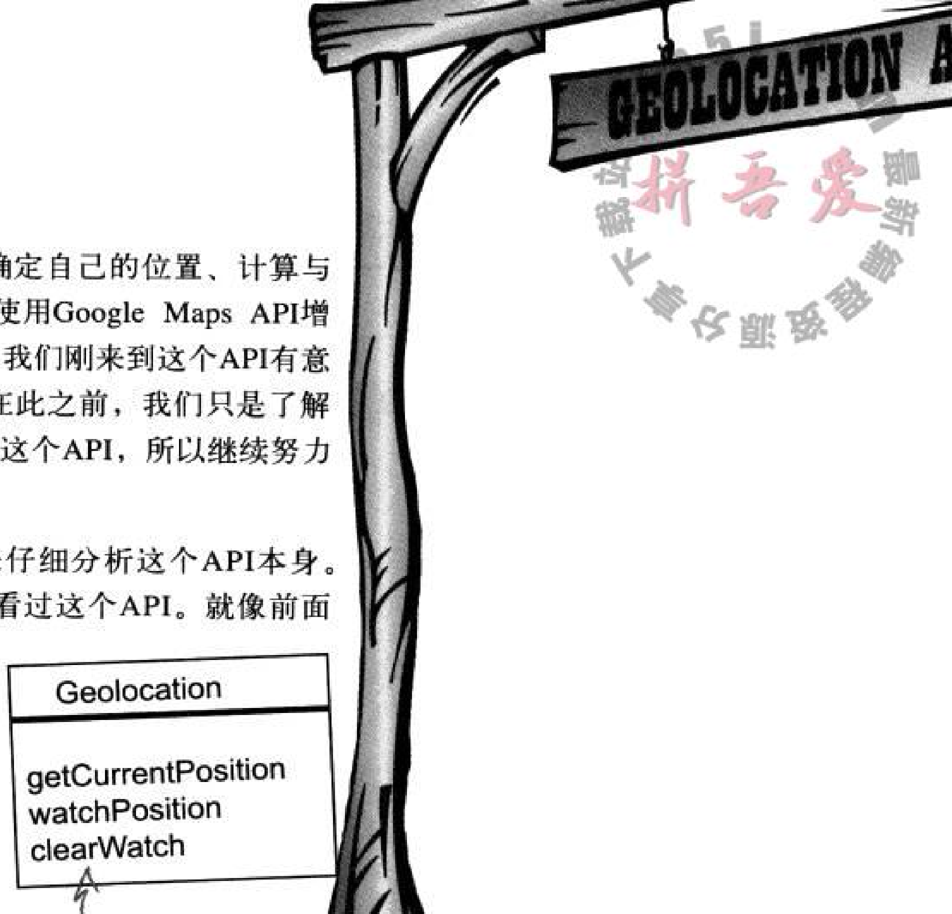
地理定位: 嗯，如果是这样，我不能保证你总能得到一个位置，不过也没关系，因为我会提供一个很好的方法来妥善处理失败。你要做的就是给我一个错误处理程序，如果我遇到问题就会调用这个错误处理程序。

Head First: 太好了。嗯，我们的时间到了。谢谢你，地理定位，感谢你来到这里，也祝贺你荣升为一个真正的W3C标准。

再来看地理定位API……

我们已经对地理定位API有些了解了，可以确定自己的位置、计算与其他位置的距离、处理API的错误状态，甚至使用Google Maps API增加了一个地图。不过现在可不是休息的时候，我们刚来到这个API有意思的部分。另外，这也是一个关键的节点，在此之前，我们只是了解这个API，而过了这个节点，就开始转为掌握这个API，所以继续努力吧！

继续前进之前，我们还需要做一件事，再来仔细分析这个API本身。我们已经讲得够多了，不过还从来没有真正看过这个API。就像前面一直说的，这个API确实相当简单，只有3个方法：`getCurrentPosition`（这个方法的作用你已经知道）、`watchPosition`（很快你就会了解）和`clearWatch`（可以猜得到，这个方法与`watchPosition`有关）。介绍这两个新方法之前，再来看一看`getCurrentPosition`和一些相关的对象，如`Position`和`Coordinates`对象。你会发现之前你不知道的一些新东西。



这些方法是地理定位API的一部分。

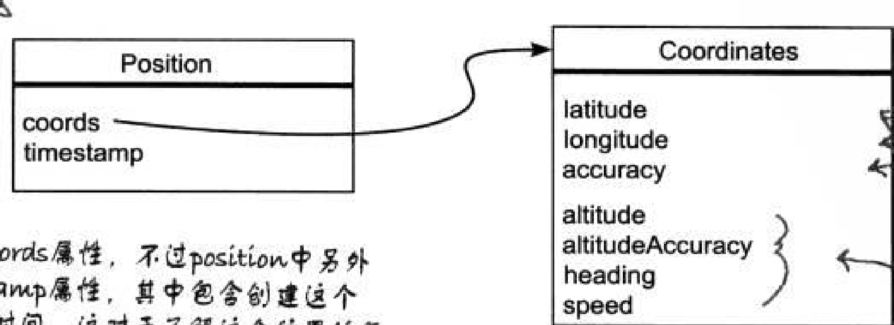
错误处理程序会在浏览器无法确定其位置时调用。前面已经看到，可能有很多原因导致无法确定位置。

```
getCurrentPosition(successHandler, errorHandler, positionOptions)
```

要记住，成功处理程序（或回调）会在确定位置时调用，并传入一个position对象。

这里还有一个没用过的参数，利用这个参数可以调整地理定位的行为。

我们知道纬度和经度，不过除了这些，coordinates对象还有其他属性。



肯定有这3个属性：纬度、经度和精度。

我们已经知道coords属性，不过position中另外还有一个timestamp属性，其中包含创建这个position对象的时间。这对于了解这个位置的年龄很有用。

其他属性有可能不支持，这取决于你的设备。

可以谈谈你的精度吗？



查找位置并不是一项精密科学。根据浏览器使用的方法，你可能只知道你所在的州、城市或街区。另外，利用更高级的设备，可能会知道精确的位置，也许误差不超过10米，还会提供你的速度、朝向和高度。

在这种情况下，我们如何编写代码呢？地理定位API的设计者与我们制订了一个很好的小约定：每次他们提供一个位置时，还会提供这个位置的精度（单位为米），可信度达到95%。所以，例如我们可能知道位置精度为500米，这说明只要考虑到半径500米，完全可以确信我们的位置落在这个半径为500米的范围内。精度为500米时，我们可以给出一些可靠的建议，比如正在哪个城市或街区，不过要提供具体街道的驾车指示可能不太合适。无论如何，显然要由你的应用来确定希望如何利用这个精度数据。

说得够多了，下面来看当前位置的精度是什么。你已经看到，精度信息是coordinates对象的一部分。下面把这部分信息拿出来，用在displayLocation函数中。

```
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var div = document.getElementById("location");

    div.innerHTML = "You are at Latitude: " + latitude + ", Longitude: " + longitude;
    div.innerHTML += " (with " + position.coords.accuracy + " meters accuracy)";

    var km = computeDistance(position.coords, ourCoords);
    var div = document.getElementById("distance");
    distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ";
    showMap(position.coords);
}
```

这里使用了position的accuracy属性，并追加到<div>的innerHTML的末尾。

试一试精度



确保把这一行代码增加到你的代码中，并加载页面。现在你可以看到位置的精度如何。一定要在你有的所有设备上都试一试。



在线测试：http://wickedlysmart.com/hfhtml5/chapter5/accuracy/myLoc.html

pin51.com 拼吾爱

“无处可逃”

这句话的出处很有争议。有些人说真正第一次提到它是在电影《Buckaroo Banzai》中，另外一些人却说它出自禅宗佛经，还有一些人从不同的书、电影和流行歌曲中找到它的出处。不管来源是哪里，总之这里会用到它，甚至在这一章之后还会更多地使用，因为我们要把它变成一个小Web应用，名字就叫做“无处可逃”。没错，真的有这样的一个应用！不过，我们还需要你（读者）的一点参与，你必须抬起屁股（请原谅我们这么说）站起来，到处走走。

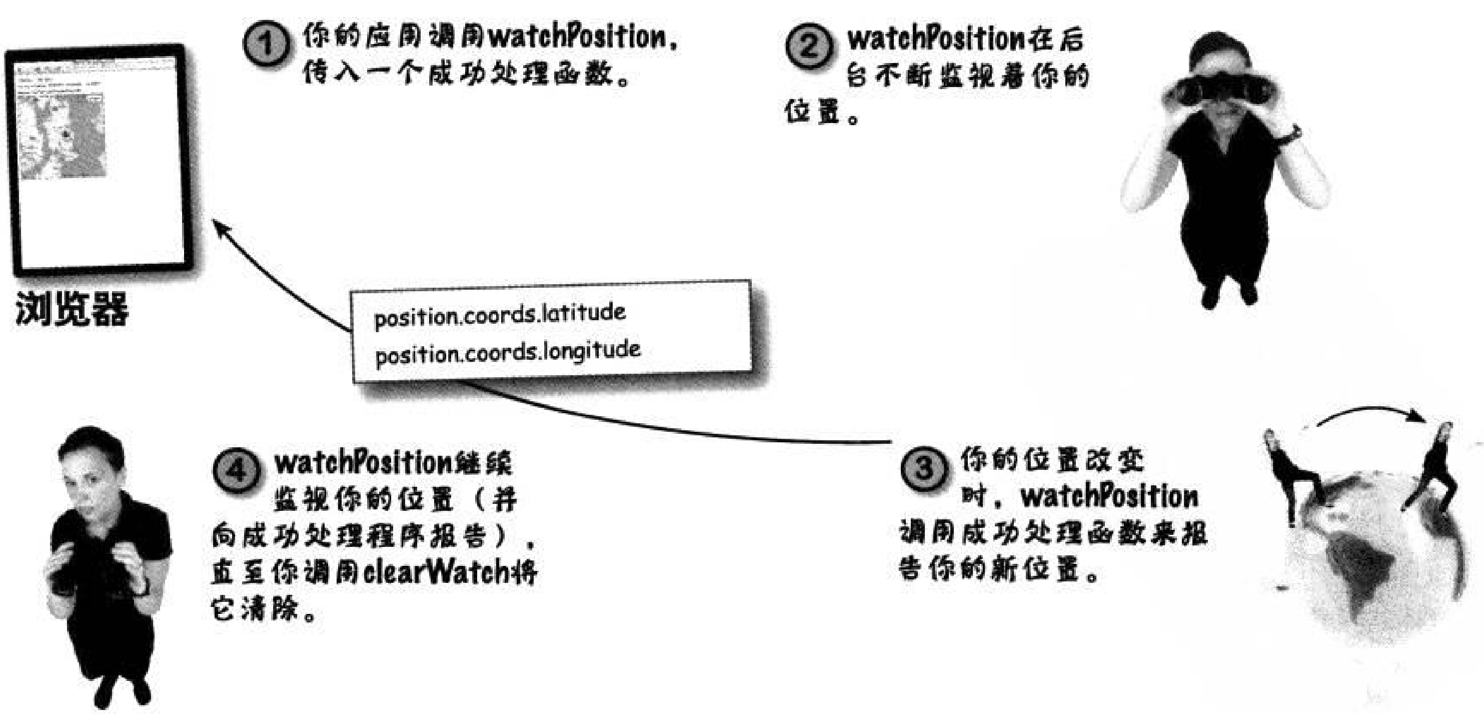


在这场争论中你站在哪一边？是出自Banzai Institute，还是源于禅宗佛经？

我们要做的就是扩展当前的代码，让它能实时跟踪你的移动。为此，我们要把所有内容集成起来，包括地理定位API的后两个方法，创建一个可以实时跟踪你的应用。

如何跟踪你的移动

前面已经提醒过，地理定位API有一个watchPosition方法。顾名思义，这个方法会监视你的移动，并在位置改变时向你报告位置。watchPosition方法看上去确实与getCurrentPosition方法很像，不过行为稍有不同，每次你的位置改变时它会重复调用你的成功处理程序。下面来看它是如何做的。



启动应用

我们要以前面的代码作为起点，首先向HTML增加几个按钮，从而能开始和结束跟踪你的位置。为什么需要这些按钮呢？嗯，首先，用户并不想一直被跟踪，他们通常希望对此有些控制。不过除此以外还有一个原因：不停地检查你的位置对于移动设备来说是一个相当耗电的操作，如果一直打开跟踪，这会严重影响你的电池寿命。所以，首先，我们将更新HTML，增加一个表单和两个按钮：一个用来监视你的位置，另一个用来停止监视。



实时跟踪用户可能非常耗电。一定要为用户提供信息，指出目前正在跟踪，另外还要提供相应的一些控件。

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Wherever you go, there you are</title>
  <script src="myLoc.js"></script>
  <link rel="stylesheet" href="myLoc.css">
</head>
<body>
  <form>
    <input type="button" id="watch" value="Watch me">
    <input type="button" id="clearWatch" value="Clear watch">
  </form>
  <div id="location">
    Your location will go here.
  </div>
  <div id="distance">
    Distance from WickedlySmart HQ will go here.
  </div>
  <div id="map">
  </div>
</body>
</html>
```

我们增加了一个表单元素，其中包括两个按钮，一个用来启动监视，id为“watch”，另一个用来清除监视，id为“clearWatch”。

我们要重用原来的<div>报告实时位置信息。

稍后还会回来考虑Google地图……

pin51.com 拼吾爱 最新免费资源下载站

调整原来的代码……

现在需要为这两个按钮增加点击处理程序。只有在支持地理定位的情况下我们才会在getMyLocation函数中增加按钮点击处理程序。另外，由于要用这两个按钮控制所有地理定位跟踪，所以要从getMyLocation删除现在的getCurrentPosition调用。下面删除这个代码，再增加两个处理程序：watchLocation对应监视按钮，clearWatch对应清除按钮：

```
function getMyLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(displayLocation,displayError);
        var watchButton = document.getElementById("watch");
        watchButton.onclick = watchLocation;
        var clearWatchButton = document.getElementById("clearWatch");
        clearWatchButton.onclick = clearWatch;
    }
    else {
        alert("Oops, no geolocation support");
    }
}
```

（代码中 `navigator.geolocation.getCurrentPosition(displayLocation,displayError);` 一行被划掉。）

如果浏览器支持地理定位，则我们会增加按钮点击处理程序。如果不支持地理定位，则增加这些处理程序毫无意义。

要调用watchLocation来启动监视，调用clearWatch停止监视。

编写watchLocation处理程序

目前，我们要做的工作是：用户点击监视按钮时，他们希望开始跟踪他们的位置。所以我们将使用geolocation.watchPosition方法开始监视他们的位置。geolocation.watchPosition方法有两个参数，一个成功处理程序和一个错误处理程序，所以我们将使用原来已有的两个处理程序。它还会返回一个watchId，可以在任何时刻使用这个id取消监视行为。我们把这个watchId放在一个全局变量中，为清除按钮编写点击处理程序时会用到这个变量。以下是watchLocation函数和watchId的代码，把这些代码增加到myLoc.js：

```
var watchId = null;

function watchLocation() {
    watchId = navigator.geolocation.watchPosition(displayLocation,
                                                   displayError);
}
```

在文件最上面增加watchId作为一个全局变量。我们把它初始化为null。以后还需要这个变量清除监视。

调用watchPosition方法，传入前面编写的成功处理程序displayLocation和现有的错误处理程序displayError。

www.minxue.net 拼吾爱 最新最全的下载资源站

编写clearWatch处理程序

现在来编写处理程序清除监视行为。为此需要得到watchId，并把它传递到geolocation.clearWatch方法。

```
function clearWatch() {
    if (watchId) {
        navigator.geolocation.clearWatch(watchId);
        watchId = null;
    }
}
```

确保有一个watchId，然后……

……调用geolocation.clearWatch方法，传入这个watchId。这会停止监视。

还需要对displayLocation做一个小小的更新……

还需要做一个很小的修改，涉及前面编写的Google Maps代码。在这个代码中，我们调用了showMap来显示Google Map。showMap会在页面中创建一个地图，这件事你只希望做一次。不过，要记住，开始用watchPosition监视你的位置时，每次位置有更新时都会调用displayLocation。

要确保只调用一次showMap，首先查看这个地图是否存在，如果不存在，则调用showMap。否则，说明showMap已经调用过（而且已经创建了地图），所以不需要再调用这个函数了。

```
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;

    var div = document.getElementById("location");
    div.innerHTML = "You are at Latitude: " + latitude + ", Longitude: " + longitude;
    div.innerHTML += " (with " + position.coords.accuracy + " meters accuracy)";

    var km = computeDistance(position.coords, ourCoords);
    var distance = document.getElementById("distance");
    distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ";

    if (map == null) {
        showMap(position.coords);
    }
}
```

如果还没有调用showMap，则调用这个函数，否则不需要每次调用displayLocation时都调用showMap。

pin51.com 拼吾爱 最新最全资源分享下载站

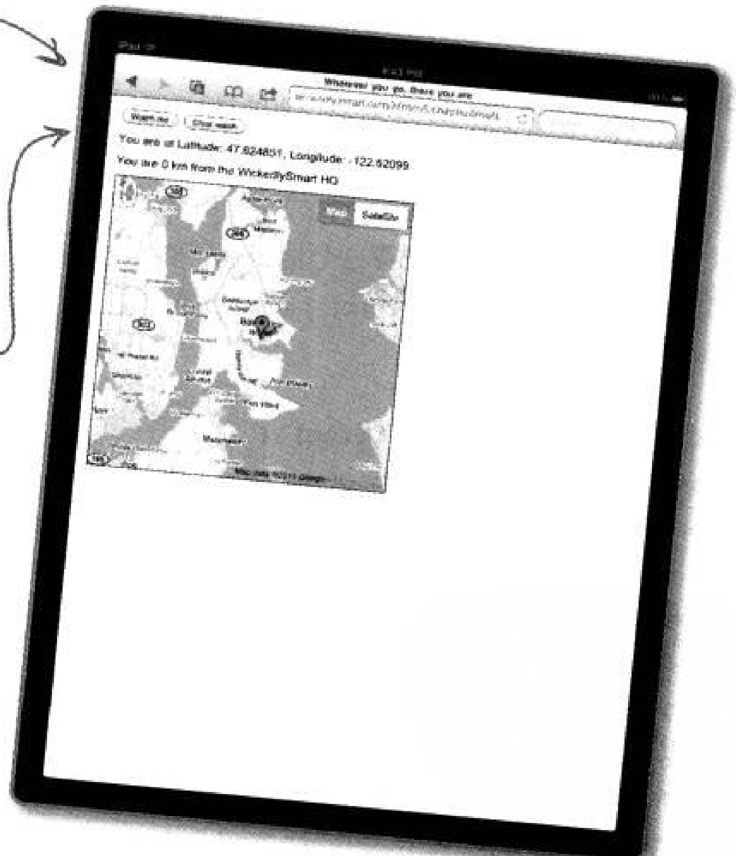
动起来！

确保已经键入所有新代码，并重新加载你的页面myLoc.html。现在，要想真正测试这个页面，还需要“重新定位”，更新你的位置。所以你可以散步、骑车、开车，或者使用你喜欢的任何交通方式。

不言而喻，如果在你的桌面浏览器上运行这个应用，肯定很没意思（因为你不能带着它移动），所以完成这个测试确实需要使用一个移动设备。另外，如果需要帮助，则希望利用你的移动设备访问一个托管版本，很高兴地告诉你，我们已经把这个代码的一个副本放在

http://wickedlysmart.com/hfhtml5/chapter5/watchme/myLoc.html

这是我们的测试运行情况……



随着你四处移动，这些数会更新。

注意，目前这个地图一直在你的初始位置居中……

在线测试：http://wickedlysmart.com/hfhtml5/chapter5/watchme/myLoc.html

there are no Dumb Questions

问： 使用watchPosition时，我怎么控制浏览器以什么频率提供位置更新呢？

答： 这是不能控制的。浏览器会确定最优的更新频率，并确定你什么时候改变了位置。

问： 我第一次加载页面时，尽管我只是静静坐着，为什么我的位置会改变好几次？

答： 要记住，我们说过浏览器可能使用好几种方法来确定你的位置，对不对？根据浏览器确定位置所用的方法（或多种方法），位置的精度会随时间改变。一般来讲，精度会越来越高，不过有时（比方如，你开车进入郊区，那里只有一个电话基站），精度也可能变糟。不管怎样都能使用position.coords对象中的accuracy属性来监视精度。

问： 我能使用coordinates对象的altitude和altitudeAccuracy属性吗？

答： 这些属性不一定会得到支持（显然，只在一些高端移动设备上支持），所以必须确保代码能处理不支持这些属性的情况。

问： 朝向（heading）和速度（speed）是什么？

答： 朝向就是你行进的方向，速度是你移动的快慢。假设你正在第5洲际公路上以55m/h的速度向北驾车行驶。那么你的朝向就是北，速度是55m/h。如果你坐在车里，而车停在星巴克咖啡馆的停车场，那么你的速度就是0，而且没有朝向（因为你没有移动）。

问： 在地图上确定我的位置到你的位置之间的距离时，比这个应用报告的距离长得多，为什么呢？

答： 要记住，我们的距离函数只是计算"笔直"的距离，而地图工具很可能会给出行驶距离。

Sharpen your pencil

下面给出displayLocation的一个候选实现。你能猜出它做什么吗？仔细看一看，在下面写出你的答案。如果你爱冒险，还可以动手试一试！

```
distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ";
if (km < 0.1) {
  distance.innerHTML = "You're on fire!";
} else {
  if (prevKm < km) {
    distance.innerHTML = "You're getting hotter!";
  } else {
    distance.innerHTML = "You're getting colder……";
  }
}
prevKm = km;
```

在这里写出这个代码做什么。 → ....

你有一些选项……

到目前为止，我们一直都避开了getCurrentPosition（和watchPosition）的第三个参数：positionOptions参数。利用这个参数，可以控制地理定位如何计算它的值。下面来看这3个选项及其默认值：

```
var positionOptions = {
    enableHighAccuracy: false,
    timeout: Infinity,
    maximumAge: 0
}
```

首先有一个属性启用高精度，稍后会介绍这是什么意思……

timeout选项会控制浏览器确定位置的时间。默认地，这设置为infinity，表示浏览器可以用所需的任意时间来得到位置。

可以把这个选项重置为一个值，单位为毫秒，比如10000，这会给浏览器10秒的时间来查找位置，否则如果在这个时间内没有找到位置，就会调用错误处理程序。

最后，maximumAge选项设置了一个位置的最大“年龄”，超过这个年龄后浏览器需要重新计算位置。默认地，这个选项值为0，表示浏览器总是要重新计算位置（每次调用getCurrentPosition时都要重新计算）。

能再谈谈你的精度吗？



我们已经见过，地理定位API交给我们的每个位置都有一个精度属性。不过，还可以告诉地理定位API：我们想要它能得到的最精确的结果。目前，这对于浏览器只作为一个提示，实际上，不同的实现可能会对这个提示做不同的处理。尽管这个选项听上去不太重要，但它确实有很大影响。例如，如果你不关心结果是不是超级精确，你可能只想知道用户在巴尔的摩—地理定位API可以相当快地告诉你，而且代价很小（从耗电角度讲）。另一方面，如果你需要知道你的用户在哪条街上，这也没问题，不过地理定位API必须启用GPS，耗费很多电来得到这个信息。利用enableHighAccuracy选项，你可以告诉地理定位API，尽管代价很大，但你确实需要所能得到的最精确的位置。要记住，使用这个选项并不能保证浏览器总能提供一个更精确的位置。

www.pin5i.com 拼吾爱

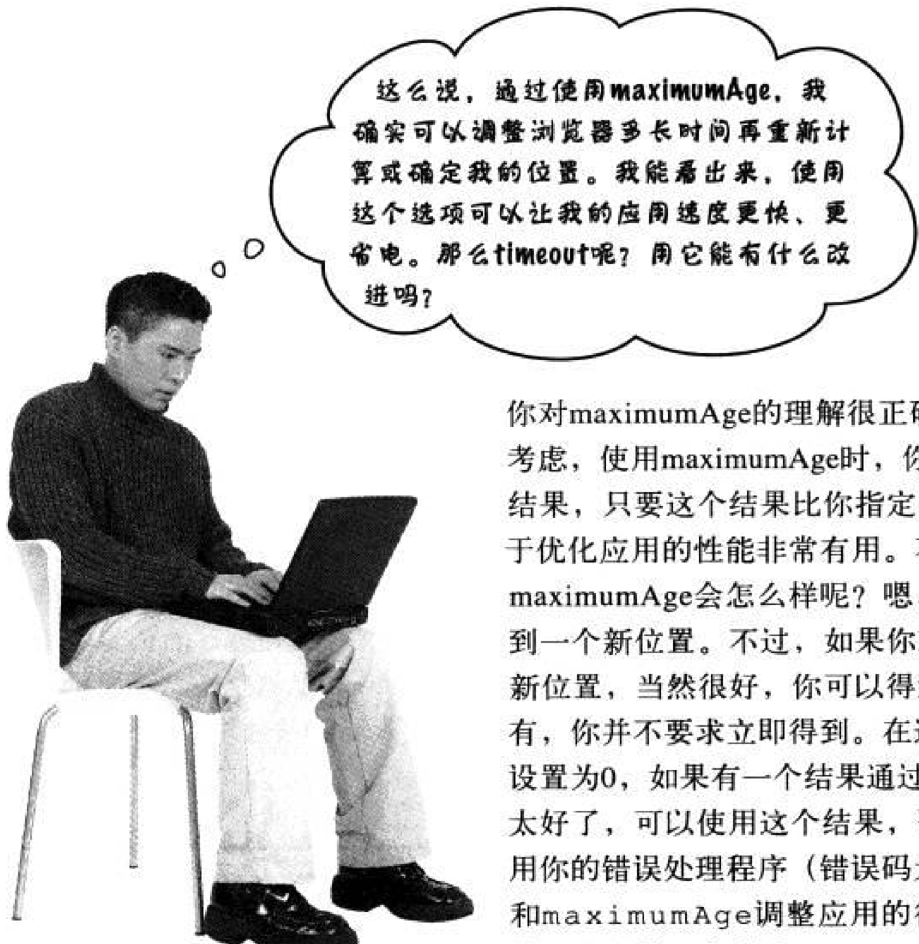
超时和最大年龄……



下面再来考虑timeout和maximumAge选项是什么：

timeout: 这个选项告诉浏览器确定用户的位置要多长时间。需要指出，如果提示用户批准一个位置请求，在用户接受之前timeout计时不会开始。如果浏览器在timeout指定的时间内（毫秒数）不能确定一个位置，会调用错误处理程序。默认地，这个选项会设置为*Infinity*。

maximumAge: 这个选项告诉浏览器一个位置可以有多老。所以，如果浏览器有一个在60秒前确定的位置，而maximumAge设置为90000（90秒），getCurrentPosition调用会返回现有的已缓存的位置（浏览器不会尝试去得到一个新位置）。不过，如果maximumAge设置为30秒，就会要求浏览器确定一个新位置。



你对maximumAge的理解很正确。对于timeout，可以这样考虑，使用maximumAge时，你会得到一个老的（缓存的）结果，只要这个结果比你指定的maximumAge年轻，这对于优化应用的性能非常有用。不过，如果位置的年龄超过maximumAge会怎么样呢？嗯，浏览器会放弃，努力去得到一个新位置。不过，如果你对此不太关心。如果有一个新位置，当然很好，你可以得到这个位置。不过，如果没有，你并不要求立即得到。在这种情况下，可以把timeout设置为0，如果有一个结果通过了maximumAge测试，那么太好了，可以使用这个结果，否则调用会立即失败，并调用你的错误处理程序（错误码为TIMEOUT）。用timeout和maximumAge调整应用的行为可以有很多有创意的方法，这只是一个例子。

pin51.com 拼吾爱

WHO DOES WHAT?

下面可以看到地理定位API的一些选项。对于每个选项，请与它的行为连连看。

`{maximumAge:600000}`	我只想要年龄小于10分钟的缓存位置。如果没有小于10分钟的缓存位置，则我会请求一个新位置，不过前提是在1秒之内就能得到。
`{timeout:1000, maximumAge:600000}`	如果浏览器有一个年龄小于10分钟的缓存位置，则我会使用这个位置，否则，我想要一个全新的位置。
`{timeout:0, maximumAge:Infinity}`	我只想要全新的位置。浏览器随便用多长时间，只要能给我一个新位置就行。
`{timeout:Infinity, maximumAge:0}`	我只想要缓存的位置。不论年龄多大。如果根本没有缓存的位置，我就会调用错误处理程序。不要给我新位置！我会离线使用。

mmm51.com 最新下载站 拼吾爱 分享资源

如何指定选项

JavaScript的好处之一是，如果我们想在一个对象中指定一组选项，可以直接在方法调用中将选项键入一个字面量对象。具体做法如下：假设我们希望启用高精度，另外把位置的最大年龄设置为60秒（或60000毫秒）。可以如下创建选项：

```
var options = {enableHighAccuracy: true, maximumAge: 60000};
```

看到了吧？JavaScript确实很能干，嗯，至少我们这样认为。☺

然后把options传递到getCurrentPosition或watchPosition，如下所示：

```
navigator.geolocation.getCurrentPosition(
                    displayLocation,
                    displayError,
                    options);
```

在这里，使用options变量传递我们的选项。

或者，也可以内联编写options对象，如下所示：

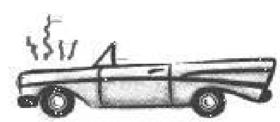
```
navigator.geolocation.getCurrentPosition(
                    displayLocation,
                    displayError,
                    {enableHighAccuracy: true, maximumAge: 60000});
```

可以看到这个技术会在JavaScript代码中大量使用。

这些是选项，在函数调用中写为一个字面量对象！有些人认为这个代码更简单，更可读。

既然已经知道了选项，了解了它们做什么，以及如何指定选项，下面就来使用这些选项。接下来我们会使用选项，不过要记住，这些选项要用来调整你的应用，而应用会有自己特有的需求。这些选项还会受你的设备、浏览器实现和网络的影响，所以你需要自己好好研究研究。

试一试诊断检查

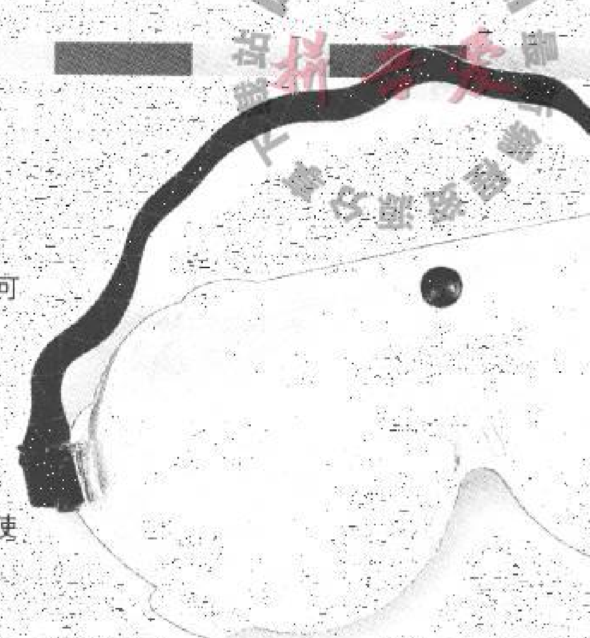


前面运行诊断时，你遇到过这种情况吗？你一直在等啊等，可是什么也没有发生！这很可能是因为超时设置为无限长。换句话说，只要浏览器没有遇到错误条件，它就会一直等待得到一个位置。嗯，现在你应该知道如何修正这个问题，因为我们可以通过设置timeout，强制地理定位API更务实一些。做法如下：

```
function watchLocation() {
    watchId = navigator.geolocation.watchPosition(
                    displayLocation,
                    displayError,
                    {timeout:5000});
}
```

通过设置timeout为5000毫秒（5秒），可以确保浏览器不会一直尝试得到一个位置。

可以试一试，你可以随意地调整选项值。



~~不要~~大胆尝试
(让地理定位充分施展)

可以查看你的浏览器能多快找到你的位置，这是不是很有趣？我们可以尽可能给浏览器增加些难度：

- 让它启用高精度。
- 不允许它使用缓存（设置maximumAge为0）。
- 先设置timeout选项为100，然后每次失败时再增加timeout。

警告：我们不知道是否所有设备和它们的电池都支持这样做，所以使用时风险自负！

初始的选项设置如下：

```
{enableHighAccuracy: true, timeout:100, maximumAge:0}
{enableHighAccuracy: true, timeout:200, maximumAge:0}
{enableHighAccuracy: true, timeout:300, maximumAge:0}
```

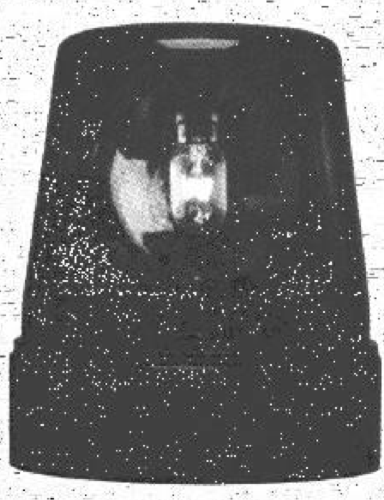
← 从这些选项开始……

← 如果失败，再试一次……

← 依此类推……

现在检查下一页的代码，你会发现这很有意思。输入这些代码，可以把它直接增加到你的myLoc.js的JavaScript代码中。在你的各种设备上试一试，把结果记录在这里：

这里写设备。 ↓ 这里写时间。



在 __________ 上找到位置需要________________ 毫秒

在 __________ 上找到位置需要________________ 毫秒

在 __________ 上找到位置需要________________ 毫秒

在 __________ 上找到位置需要________________ 毫秒

在线测试：http://wickedlysmart.com/hfhtml5/chapter5/speedtest/speedtest.html

```
var options = { enableHighAccuracy: true, timeout:100, maximumAge: 0 };
window.onload = getMyLocation;
function getMyLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(
            displayLocation,
            displayError,
            options);
    }  else {
        alert("Oops, no geolocation support");
    }
}
function displayError(error) {
    var errorTypes = {
        0: "Unknown error",
        1: "Permission denied",
        2: "Position is not available",
        3: "Request timeout"
    };
    var errorMessage = errorTypes[error.code];
    if (error.code == 0 || error.code == 2) {
        errorMessage = errorMessage + " " + error.message;
    }
    var div = document.getElementById("location");
    div.innerHTML = errorMessage;
    options.timeout += 100;
    navigator.geolocation.getCurrentPosition(
        displayLocation,
        displayError,
        options);
    div.innerHTML += " ...... checking again with timeout=" + options.timeout;
}
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var div = document.getElementById("location");
    div.innerHTML = "You are at Latitude: " + latitude +
                    ", Longitude: " + longitude;
    div.innerHTML += " (found in " + options.timeout + " milliseconds)";
}
```

首先初始化选项，timeout为100，maximumAge为0。

这里与往常一样，displayLocation和displayError作为成功和错误处理程序，另外传入options作为第三个参数。

首先给出错误处理程序。

这里的代码是一样的……

不过，如果出现失败，我们会把timeout选项增加100ms，然后再试。另外还会让用户知道我们在重试。

浏览器成功得到你的位置时，我们会让用户知道花了多长时间。

完成这个应用！

坐下来，好好想想，只需要一点点HTML和JavaScript，你就已经创建了一个Web应用，它不仅能确定你的位置，还能几乎实时地跟踪和显示位置。哇呜！真是士别三日，当刮目相看，HTML（当然还有你的水平）真是不可同日而语了！

不过，说到这个应用，你难道不认为还需要再“抛光”一下才能算完成吗？例如，你移动时，可以在地图上显示你的位置，另外还可以更进一步显示你原来曾在哪里，这样就能在地图上创建一个路径。

下面编写一个函数，在你移动时，保持地图以你的位置居中，每次到达一个新位置时放置一个新标记：



OK，我们把这个函数叫做scrollMapToPosition，并向它传递一个位置的坐标。

这个坐标是你的最新位置，要将地图在这个位置居中，并在这里放置一个标记。

```
function scrollMapToPosition(coords) {
    var latitude = coords.latitude;
    var longitude = coords.longitude;
    var latlong = new google.maps.LatLng(latitude, longitude);

    map.panTo(latlong);

    addMarker(map, latlong, "Your new location", "You moved to: " +
                                latitude + ", " + longitude);
}
```

首先获得新的纬度和经度，为它们创建一个google.maps.LatLng对象。

地图的panTo方法取这个LatLng对象并滚动地图，使你的新位置位于地图中心。

最后，使用前面写的addMarker函数为这个位置增加一个标记，传入地图、LatLng对象、新标记的一个标题和一些内容。



集成我们的新函数

现在，我们只需要更新displayLocation函数，每次位置改变时让它调用scrollMapToPosition。要记住，第一次调用displayLocation时，需要调用showMap来创建地图，并为你的初始位置显示一个标记。在此之后，每次只需要调用scrollMapToPosition增加一个新的标记（无需再调用showMap），并让地图重新居中。下面给出修改后的代码：

```
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var div = document.getElementById("location");
    div.innerHTML = "You are at Latitude: " + latitude
                    + ", Longitude: " + longitude;
    div.innerHTML += " (with " + position.coords.accuracy + " meters accuracy)";
    var km = computeDistance(position.coords, ourCoords);
    var distance = document.getElementById("distance");
    distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ";

    if (map == null) {
        showMap(position.coords);
    } else {
        scrollMapToPosition(position.coords);
    }
}
```

第一次调用displayLocation时，需要画出地图，并增加第一个标记。

在这之后，只需要在现有的地图上增加一个新标记。

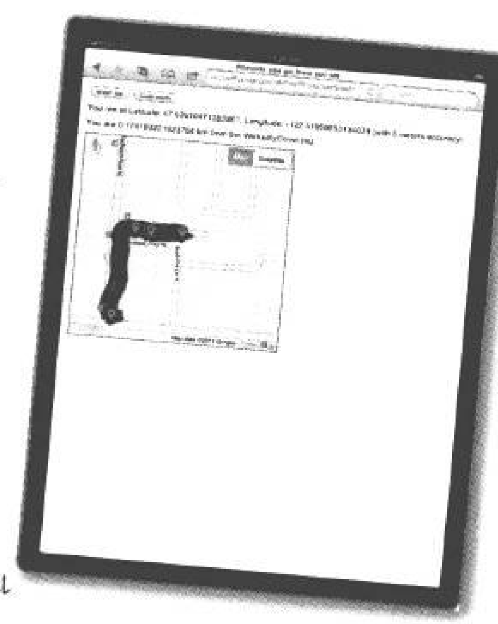
再一次……



重新加载你的页面，开始移动…… 你的地图会跟着你变化吗？你应该能看到，随着你的移动，地图上会增加一个标记轨迹（除非你一直坐在桌子前面不动）。

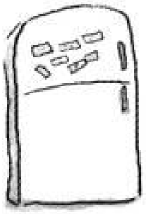
现在可以提交这个应用了，它将作为“无处可逃”的可靠证据。

这是从Wickedly Smart总部到秘密藏身地的最新路径，沿路提供了标记轨迹……噢，等等，我们不该说的……



在线测试：http://wickedlysmart.com/hfhtml5/chapter5/watchmepan/myLoc.html

pin51.com 拼吾爱 最新 下载站



代码磁贴

结束这一章之前，我们认为你可能希望这个应用更出彩些。你可能已经注意到（有些情况下），监视位置时地图上是不是增加了太多的标记？

怎么回事？watchPosition在频繁地检测移动，所以每过几步就会调用displayLocation成功处理程序。修正这个问题的一种办法是增加一些代码，这样我们必须移动某个比较大的距离，比如说用20米来测试，只有移动超过20米时才会创建一个新标记。

我们已经有了一个函数来计算两个坐标之间的距离（computeDistance），所以要做的就是每次调用displayLocation时把我们的位置保存下来，查看前一个位置和这个新位置之间的距离是否大于20米，然后再调用scrollMapToPosition。下面的一些代码会做这个工作，你的任务是完成这个代码。要当心，有些磁贴可能要用好几次！




```
var ____________________;
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var div = document.getElementById("location");
    div.innerHTML = "You are at Latitude: " + latitude + ", Longitude: " + longitude;
    div.innerHTML += " (with " + position.coords.accuracy + " meters accuracy)";
    var km = computeDistance(position.coords, ourCoords);
    var distance = document.getElementById("distance");
    distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ";
    if (map == null) {
        showMap(position.coords);
        prevCoords = ________________;
    } else {
        var meters = ________________(position.coords, prevCoords) * 1000;
        if (____________ > ______) {
            scrollMapToPosition(position.coords);
            ______________ = ________________;
        }
    }
}
```

磁贴：

`computeDistance`

`meters`

`prevCoords = null;`

`20`

`position.coords`

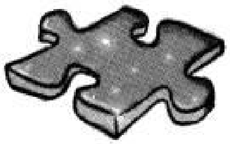
`prevCoords`

www.pin5i.com 拼吾爱 最新最全免费分享下载站

BULLET POINTS

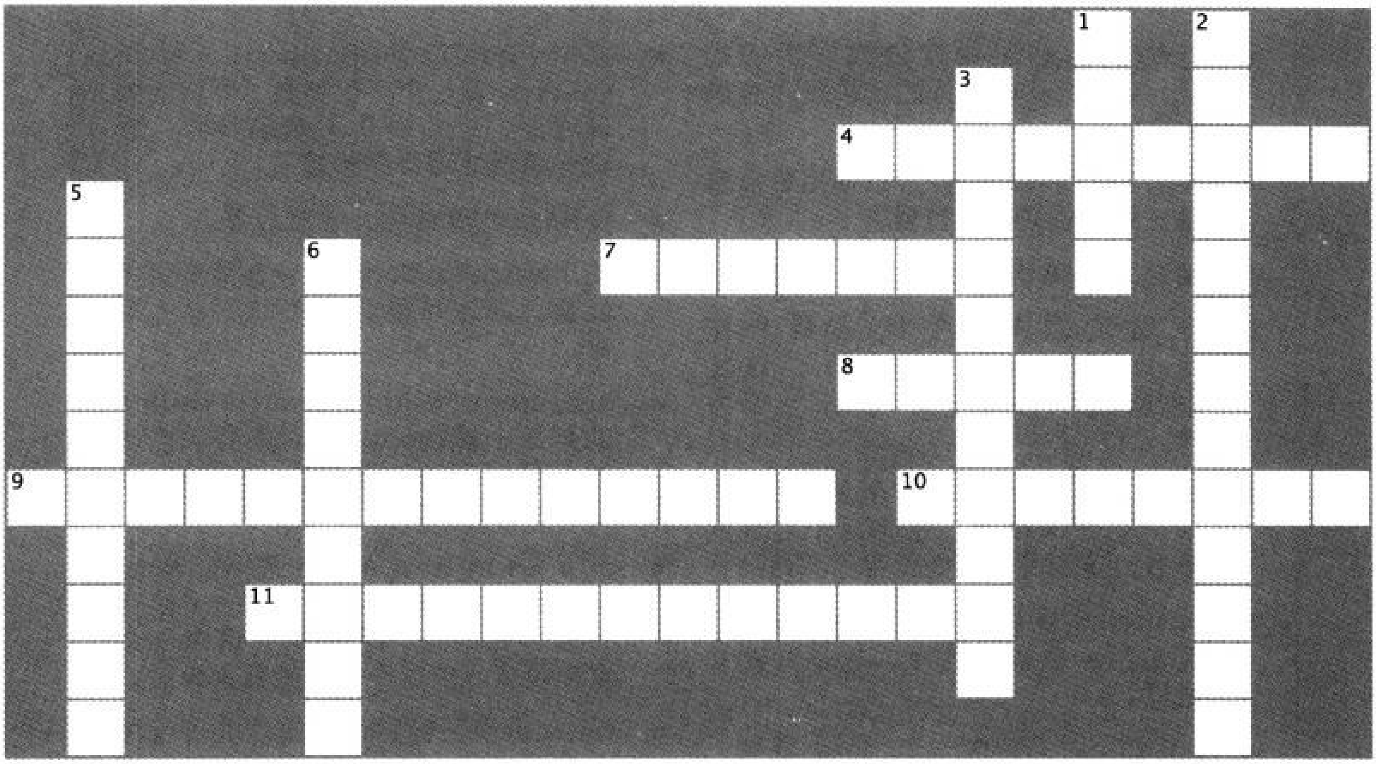
- 地理定位并不“正式”算是HTML5规范的一部分，不过可以认为它属于HTML5规范“家族”。
- 有很多方法来确定你的位置，这取决于你的设备。
- 与蜂窝基站三角定位或网络IP相比，GPS是获得位置的一种更为精确的方法。
- 没有GPS的移动设备可以使用蜂窝基站三角定位来确定位置。
- 地理定位API有3个方法和一些属性。
- 地理定位API中的主要方法是getCurrentPosition，这是navigator.geolocation对象的一个方法。
- getCurrentPosition有一个必要参数，即成功处理程序，还有两个可选的参数，分别是错误处理程序和选项。
- position对象传递到成功处理程序，其中包含位置的信息，包括纬度和经度。
- position对象包含一个coords属性，这是一个coordinates对象。
- coordinates对象的属性包括纬度（latitude）、经度（longitude）和精度（accuracy）。
- 有些设备可能还支持其他coordinates属性：高度（altitude）、高度精度（altitudeAccuracy）、朝向（heading）和速度（speed）。
- 使用accuracy属性来确定位置的精确度（单位为米）。
- 调用getCurrentPosition时，浏览器必须验证你允许共享你的位置。
- watchPosition是geolocation对象的一个方法，会监视你的位置，并在位置改变时调用一个成功处理程序。
- 类似于getCurrentPosition，watchPosition 有一个必要参数，即成功处理程序，还有两个可选的参数，分别是错误处理程序和选项。
- 使用clearWatch停止监视位置。
- 使用watchPosition时，设备需要更多能量，所以可能会缩短你的电池寿命。
- getCurrentPosition和watchPosition的第三个参数options对象有一些属性，可以设置这些属性来控制地理定位API的行为。
- maximumAge属性确定getCurrentPosition是否使用一个缓存位置，如果是，它指定了请求一个全新位置之前这个位置的最大年龄。
- timeout属性确定调用错误处理程序之前getCurrentPosition可以有多长时间来得到一个全新的位置。
- enableHighAccuracy属性向设备提供一个提示，如果可以得到一个高精度的位置，需要更多能量。
- 可以使用地理定位API并结合Google Maps API在地图上显示你的位置。

pin51.com 拼吾爱



HTML5填字游戏

这一章对第一个JavaScript API已经介绍得够多了。通过这个填字游戏把这些内容牢牢记住。



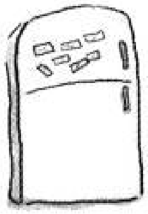
横向

4. 经度从英国____________开始测量。
7. 精度会对你的应用产生影响，因为它会影响________寿命。
8. 如果浏览器请求你共享你的位置时你拒绝了，就会调用你的错误处理程序而且________ 码为1。
9. “无处可逃”在电影______________________中曾提到。
10. 如果你的坐标_____________不好，就不要向别人提供行驶方向。
11. _________________总部的秘密位置是47.62485, -122.52099。

纵向

1. 使用__________方法将地图重新居中。
2. 没有GPS的老式设备使用蜂窝基站______________来确定你的位置。
3. __________ 的纬度和经度是40.77, -73.98。
5. 如果你把____________设置为0，就不会得到一个缓存的位置。
6. 可以使用 ______________ 公式来找到两个坐标之间的距离。

www.pin5i.com 拼吾爱 最新最全下载站 分享资源

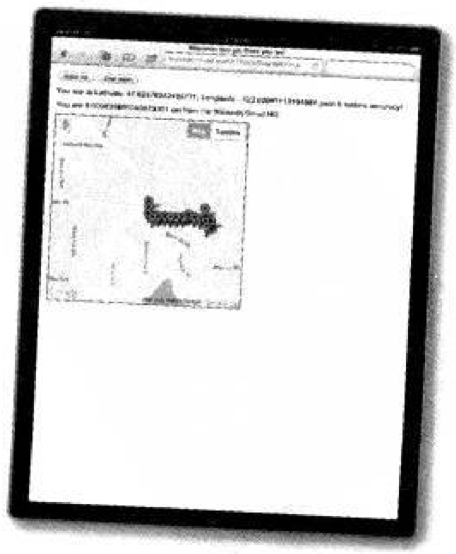


代码磁贴

你的任务是完成下面的代码，所以只有在增加上一个标记之后又走了超过20米时，我们才会显示一个新的标记。使用冰箱上的磁贴完成这个代码。当心，有些磁贴可能要用好几次！下面给出我们的答案。

```
var prevCoords = null;
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var div = document.getElementById("location");
    div.innerHTML = "You are at Latitude: " + latitude + ", Longitude: " + longitude;
    div.innerHTML += " (with " + position.coords.accuracy + " meters accuracy)";
    var km = computeDistance(position.coords, ourCoords);
    var distance = document.getElementById("distance");
    distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ";
    if (map == null) {
        showMap(position.coords);
        prevCoords = position.coords;
    }
    else {
        var meters = computeDistance(position.coords, prevCoords) * 1000;
        if (meters > 20) {
            scrollMapToPosition(position.coords);
            prevCoords = position.coords;
        }
    }
}
```

更漂亮了！



在线测试：http://wickedlysmart.com/hfhtml5/chapter6/final/myLoc.html

pin51.com 拼吾爱

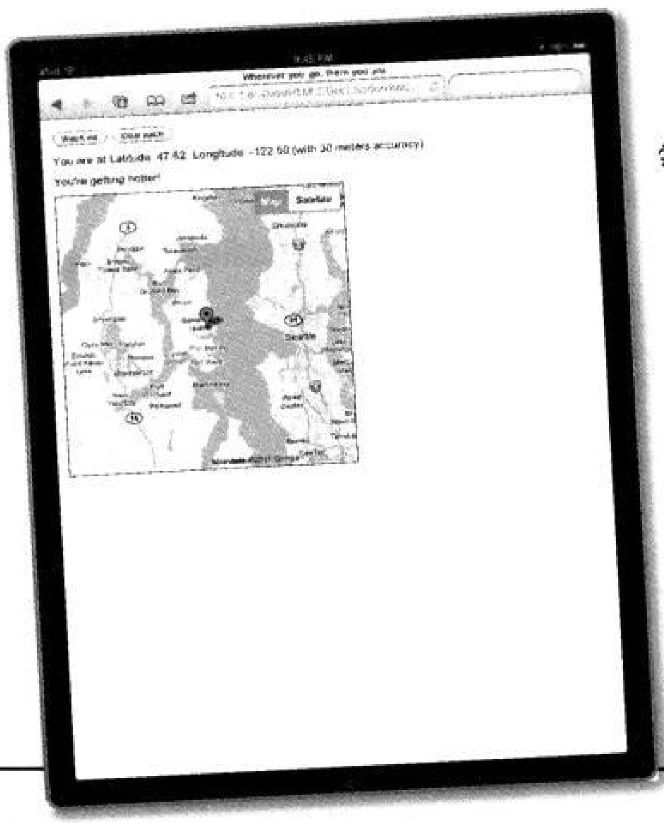
Sharpen your pencil Solution

下面给出displayLocation的一个候选实现。你能猜出它做什么吗？仔细看一看，在下面写出你的答案。如果你爱冒险，还可以动手试一试！这是我们的答案。

```
distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ";
if (km < 0.1) {
  distance.innerHTML = "You're on fire!";
} else {
  if (prevKm < km) {
    distance.innerHTML = "You're getting hotter!";
  } else {
    distance.innerHTML = "You're getting colder……";
  }
}
prevKm = km;
```

在这里写出这个代码做什么。

这个代码把我们的应用变成一个冷/热游戏。如果你离WickedlySmart总部更近它会显示“getting hotter”（更热了）消息，或者如果你越来越远离，就会显示“getting colder”（更冷了）。如果在总部的0.1千米范围内，消息会变成“You're on fire”（你要着火了）。



尝试运行这个应用，这是我们得到的结果！

拼吾爱

WHO DOES WHAT? 答案

下面可以看到地理定位API的一些选项。对于每个选项，请与它的行为连连看。

`{maximumAge:600000}`

我只想要年龄小于10分钟的缓存位置。如果没有小于10分钟的缓存位置，我会请求一个新位置，不过前提是在1秒之内就能得到。

`{timeout:1000, maximumAge:600000}`

如果浏览器有一个年龄小于10分钟的缓存位置，我会使用这个位置，否则，我想要一个全新的位置。

`{timeout:0, maximumAge:Infinity}`

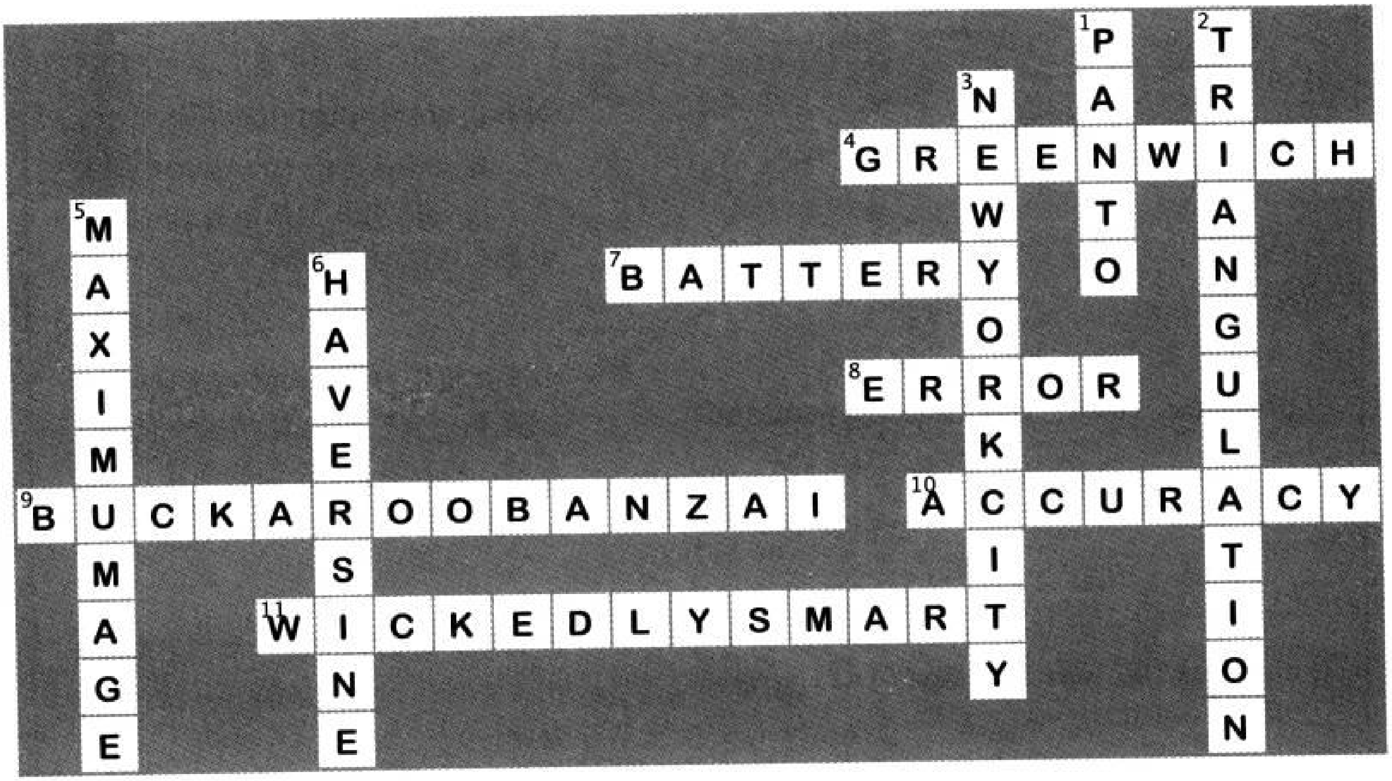
我只想要全新的位置。浏览器随便用多长时间，只要能给我一个新位置就行。

`{timeout:Infinity, maximumAge:0}`

我只想要缓存的位置。不论年龄多大。如果根本没有缓存的位置，我会调用错误处理程序。不要给我新位置！我会离线使用。

pin51.com 拼吾爱 最新最热资源分享下载站

HTML5填字游戏答案



pin51.com 拼吾爱 最新最全资源下载站

喜欢社交的应用



你留在页面里太久了。该出去走走，与Web服务聊聊，收集些数据带回来，这样就能构建更好的应用，把所有这些数据融合起来。这是编写现代HTML5应用很重要的一部分，不过，要做到这样，你必须知道如何与Web服务交流。这一章我们就来讨论这个内容，在你的页面中结合一个实际Web服务的数据。了解如何做到之后，你就能走出去，与你想要的任何Web服务交往了。我们甚至会告诉你同Web服务交流时要用的最流行的“行话”。所以，跟我来，你会用到更多的API：通信API。

pin51.com 拼吾爱

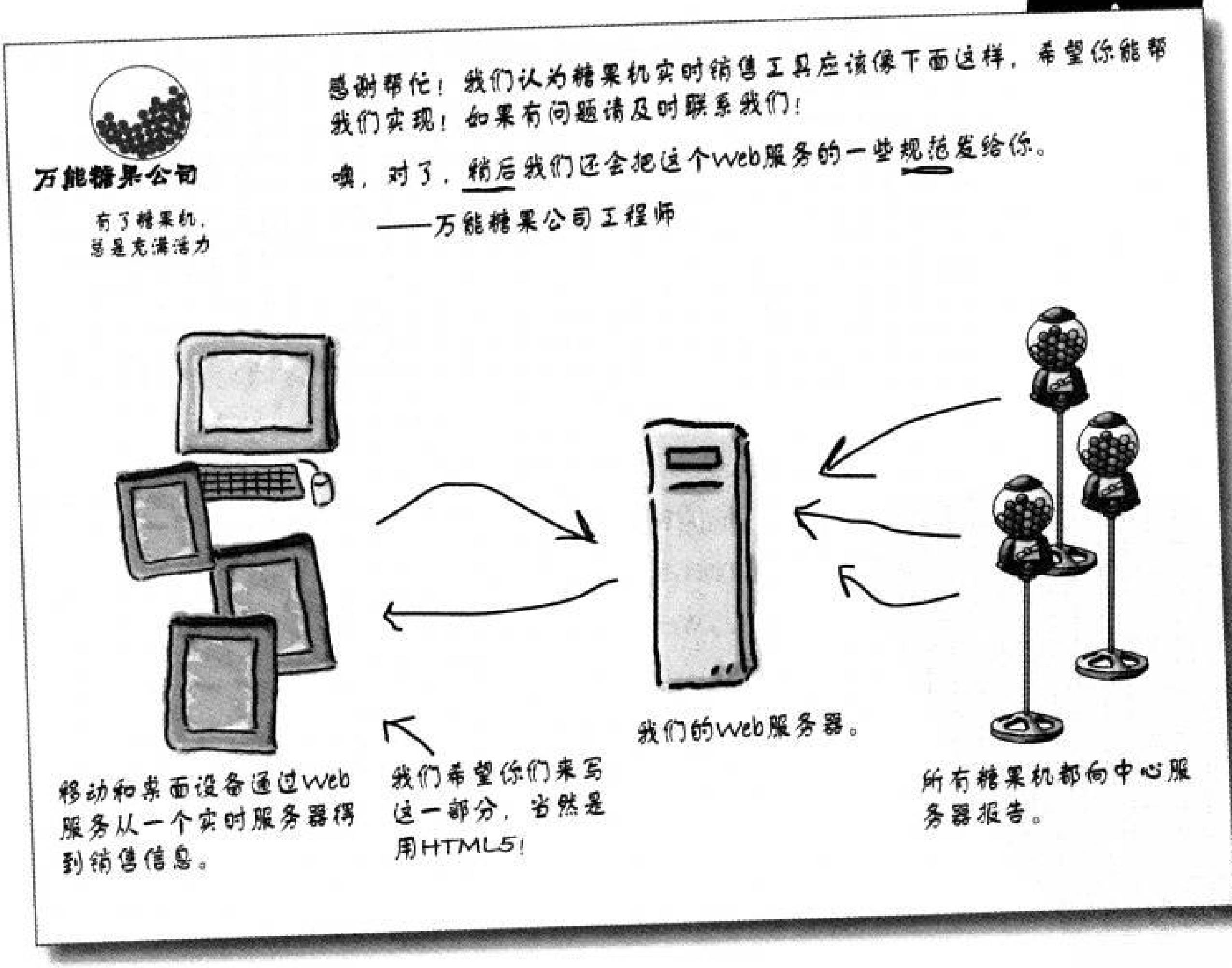
万能糖果公司需要一个Web应用

任务来了：万能糖果公司（一家构建和部署糖果机的新兴公司）与我们联系寻求帮助。可能你还不太了解他们,情况是这样的：他们最近实现了糖果机的网络支持，可以近实时地跟踪销售情况。

不用怀疑，万能糖果公司的工程师都是糖果机专家，但不是软件开发人员，所以他们希望我们帮忙构建一个应用，能帮助他们监视糖果的销售。

这是他们发来的信函：





pin51.com

开始工作之前，先花点时间想一想，如何设计应用从一个Web服务获取数据，然后根据这些数据保持Web页面不断更新。你可能还不知道如何获取数据，不用担心，现在只需要考虑高层设计。画个图，做些标记，为可能需要的代码写出伪代码。可以把这当成一个热身，让你的大脑开动起来……

设计说明

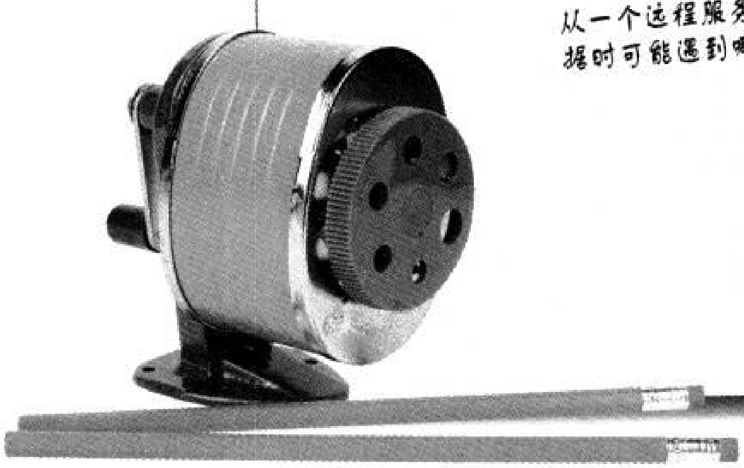
万能糖果公司

有了糖果机，总是充满活力

如何从Web服务得到数据交给我们的Web页面？

一旦得到数据，如何更新页面？

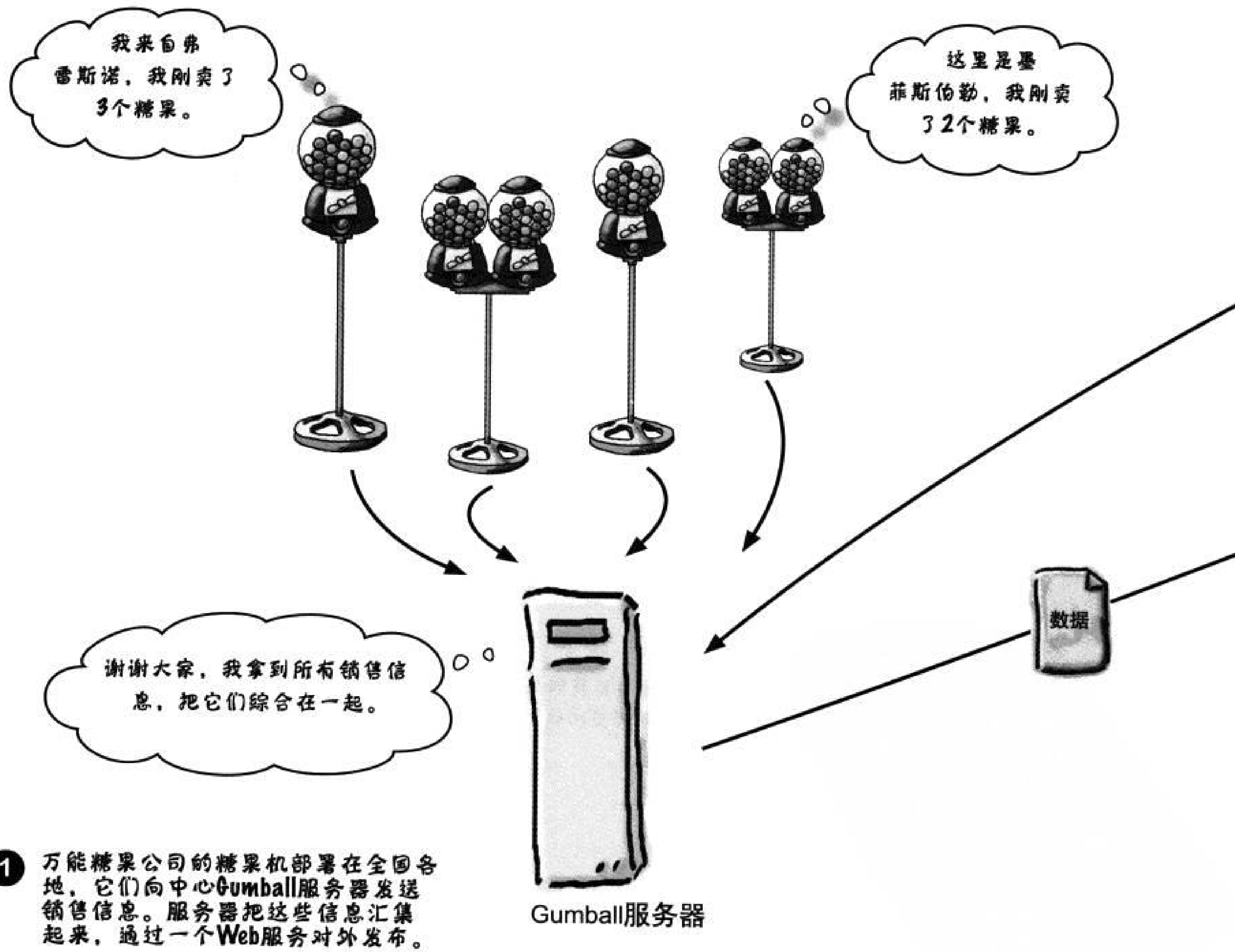
从一个远程服务器获取数据时可能遇到哪些问题？



pin51.com 拼吾爱 最新最全资源分享下载站

万能糖果公司的更多背景介绍

除了万能糖果公司的简短说明，你可能对这家公司的背景了解很少。我们了解的情况是：首先，他们让分布在全国各地的糖果机向一个万能糖果服务器发送销售报告，这个服务器会综合所有这些报告，并通过一个Web服务对外发布。另外，他们请我们构建一个Web应用，能够通过浏览器为糖果销售团队显示销售情况。他们可能非常希望当销售情况随时间改变时这个报告能够相应地更新。下面给出这个设计的高层视图：



❶ 万能糖果公司的糖果机部署在全国各地，它们向中心Gumball服务器发送销售信息。服务器把这些信息汇集起来，通过一个Web服务对外发布。

pin51.com 拼吾爱 最新资源分享下载站

❷ 浏览器加载万能糖果公司的Web应用，包括HTML标记、CSS和JavaScript。

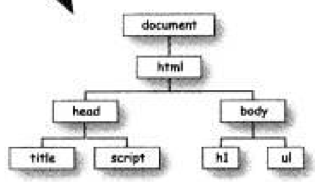




❸ 应用发出一个Web请求，从Gumball服务器获取汇集的销售信息。



❺ 应用查看数据，然后更新页面的DOM来反映新的销售数据。



❹ 应用收到Gumball服务器发回的数据。



❻ 浏览器根据DOM更新页面，用户会看到相应结果。

❼ 应用返回第3步，继续请求新数据。相应地，页面看上去在近实时地更新。

pin5i.com 拼吾爱

快速启动……

既然要等着万能糖果公司提供规范，利用这个间隙我们来做点HTML工作。

你可能有个想法，觉得我们不需要大量HTML标记来从头开始构建一个Web应用，你的想法很正确。我们只需要一个合适的地方，销售报告到来时可以把它们放在这里，其余的事情都由JavaScript来做。键入下面的代码，然后我们再来分析如何通过Web获取信息。

```
<!doctype html>
<html lang="en">
<head>
<title>Mighty Gumball (JSON)</title>
<meta charset="utf-8">
<script src="mightygumball.js"></script>
<link rel="stylesheet" href="mightygumball.css">
</head>
<body>
<h1>Mighty Gumball Sales</h1>
<div id="sales">

</div>
</body>
</html>
```

这就是标准的HTML5首部和体。

链接到一个JS文件，所以可以知道：很快我们就要编写一些JavaScript了！

建立CSS，为万能糖果公司销售报告提供样式，使CEO能看到一个漂亮的报告。

这里是占位符，用来放置销售数据。每个销售数据作为一个<div>增加到这里。

发动引擎……

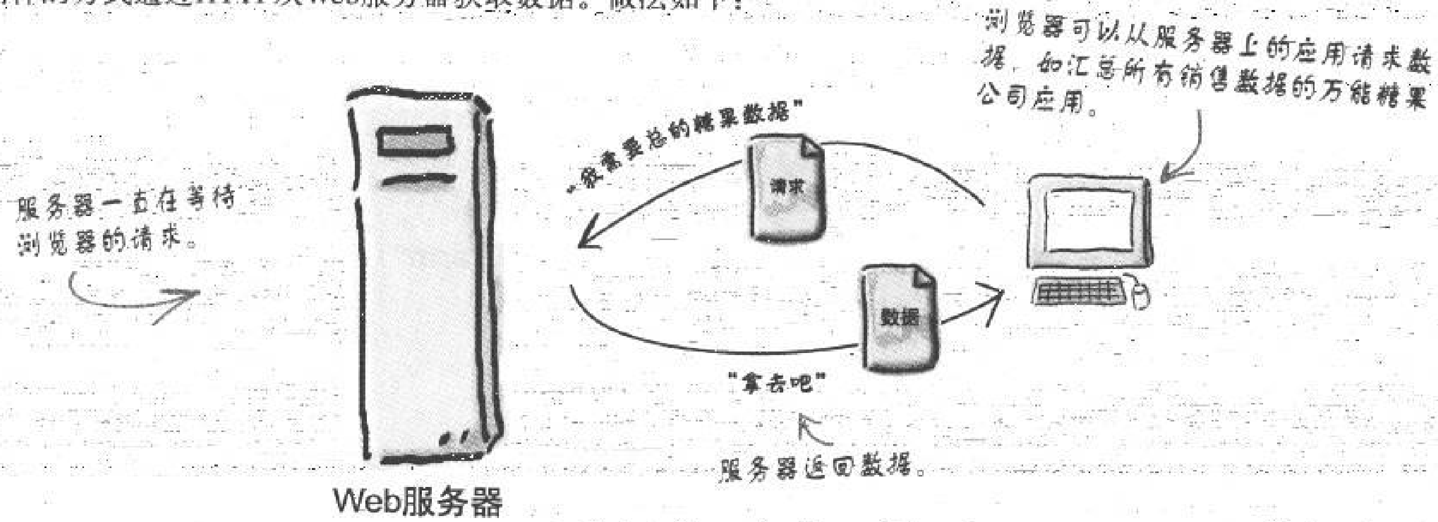


输入上面的代码，把它加载到你喜欢的浏览器，学习后面的内容之前先试一试这个代码。记住，可以从`http://wickedlysmart.com/hfhtml5`下载CSS（和本章的其他代码）。

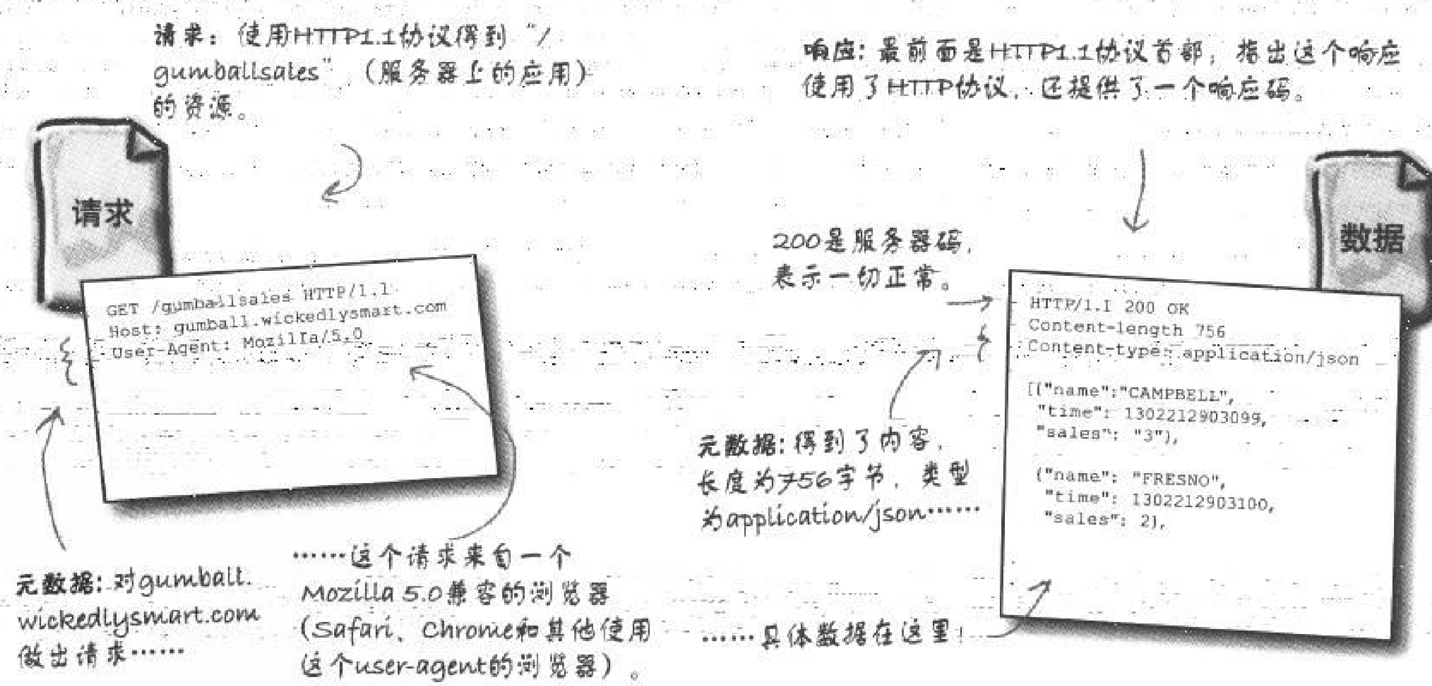
下载站 pin51.com 拼吾爱 最新资源

如何向Web服务做出请求？

下面先稍稍退一步…… 你已经知道浏览器如何向Web服务器请求页面——它向服务器做出一个HTTP请求，服务器会返回页面，并随之返回另外一些（通常）只有浏览器能看到的元数据。你可能不知道，浏览器还可以采用同样的方式通过HTTP从Web服务器获取数据。做法如下：



再仔细看一看向服务器做出的请求以及服务器返回的响应，这对我们很有帮助。请求负责告诉服务器我们想要什么数据（有时称为我们想要的"资源"），响应会包含元数据，如果一切正常，还会包含我们请求的数据：



说明：这种使用XMLHttpRequest获取数据的模式通常称为"Ajax"或XHR。

pin51.com 拼吾爱 最新最全资源分享下载站

如何从JavaScript做出请求

好了，我们知道可以用HTTP获取数据，那又怎么样呢？我们要编写一点点代码来创建一个真正的HTTP请求，然后让浏览器代表我们发出这个请求。做出请求之后，浏览器再为我们传回它接收到的数据。下面逐步分析如何做出一个HTTP请求：

❶ 首先从一个URL开始。毕竟，我们要告诉浏览器到哪里找我们想要的数据：

这是我们的URL，位于someserver.com。

"json"是指一种交换数据的格式，后面还会讨论这个内容。

```
var url = "http://someserver.com/data.json";
```

把这个URL放在一个变量url中，稍后会用到这个变量。

❷ 接下来创建一个请求对象，如下所示：

```
var request = new XMLHttpRequest();
```

将这个请求对象赋至一个变量request。

使用XMLHttpRequest构造函数创建一个新的请求对象。稍后会介绍其中的"XML"部分。



一个全新的XMLHttpRequest对象。

❸ 下面需要告诉这个请求对象我们希望它获取哪个URL，以及要使用哪种请求（与上一页的请求类似，这里将使用标准的HTTP GET请求）。为此，我们使用了请求对象的open方法。"open"方法听上去好像不仅会在请求对象中设置这些值，还会打开连接并获取数据。但实际上并非如此。尽管这个方法名有"打开"的意思，但open只是用一个URL建立一个请求，并告诉这个请求对象要使用哪种请求，以便XMLHttpRequest验证连接。可以如下调用open方法：

```
request.open("GET", url);
```

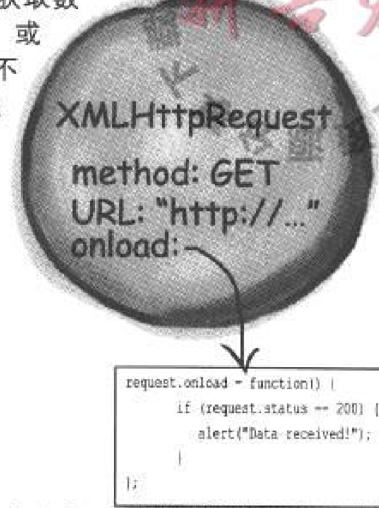
这会使用HTTP GET建立一个请求对象，HTTP GET是获取HTTP数据的标准方法。

并设置这个请求使用存储在url变量中的URL。

更新的XMLHttpRequest对象，它知道要去哪里。



4 好了，到重点了，这也是XMLHttpRequest的关键：最后要求XMLHttpRequest对象获取数据时，它会自己去获取数据。可能需要90毫秒（按计算时间来讲这只是一瞬间），或者如果速度很慢，可能要花10秒（这就是相当长的一段时间了）。所以，我们并不是一直傻傻地等待数据，而是会提供一个处理程序，数据到达时就会调用这个处理程序。如下建立处理程序（你应该很熟悉了）：



我们的请求对象。

```
request.onload = function() {
    if (request.status == 200) {
        alert("Data received!");
    }
};
```

浏览器从远程Web服务得到一个回答时，它会调用这个函数。

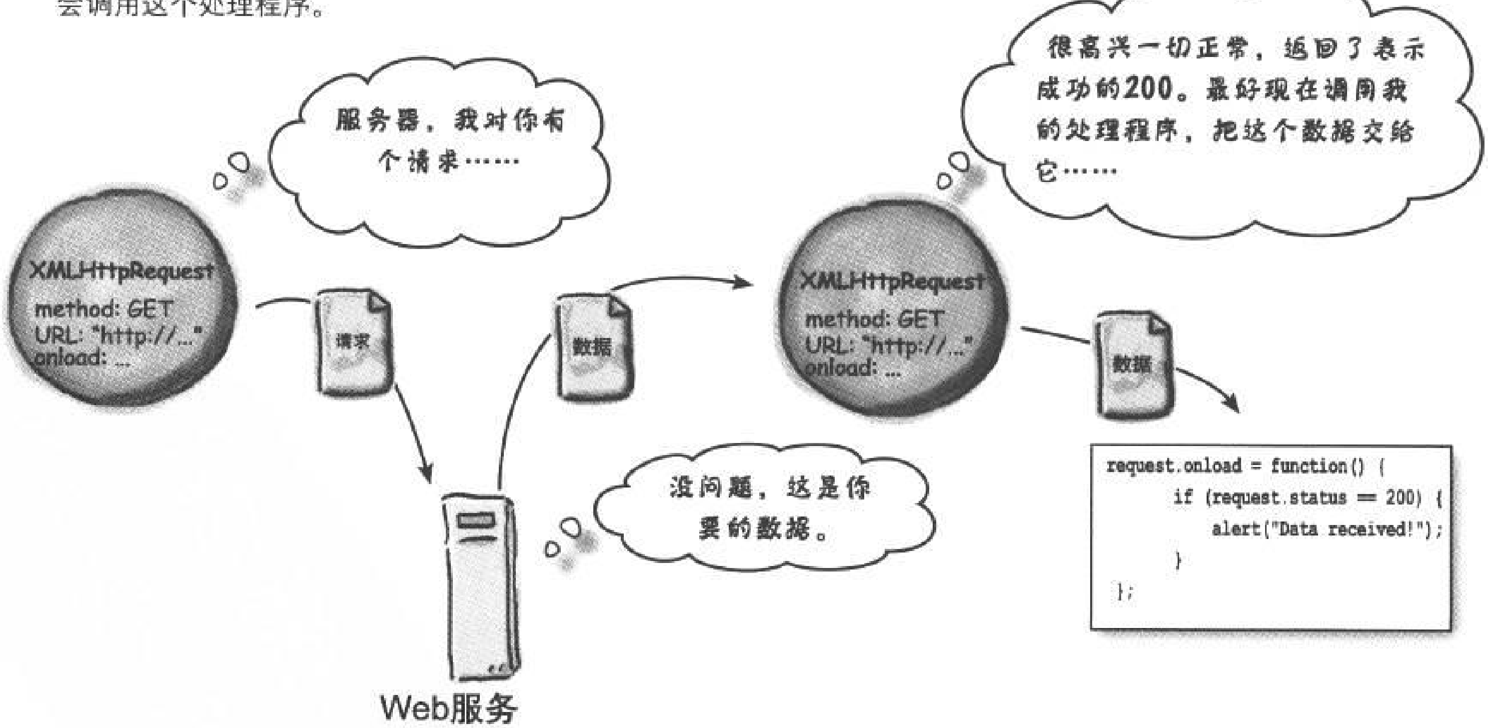
处理程序首先需要检查返回码是否为200或"OK"，然后可以对这个数据做任何处理。现在我们只是告诉用户得到了数据。后面还会编写更有意思的代码来丰富这个功能。

5 还有最后一步：需要告诉请求对象去获取数据，为此要使用send方法：

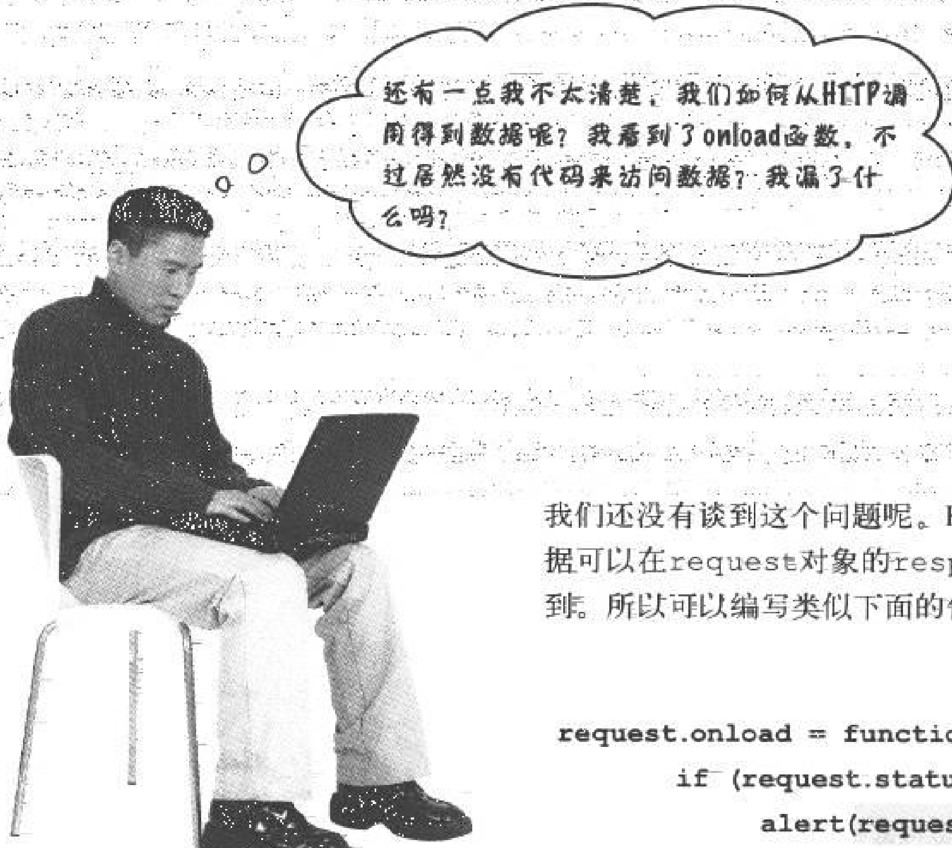
```
request.send(null);
```

这会把请求发送到服务器。如果不打算向远程服务发送任何数据，就要传入null（这里就没有发送数据）。

总结一下：我们要创建一个XMLHttpRequest对象，基于一个URL和HTTP请求类型加载这个对象，同时提供一个处理程序。然后发出请求，等待数据到达。数据到达时，就会调用这个处理程序。



pin51.com 拼吾爱



我们还没有谈到这个问题呢。HTTP GET获取的数据可以在`request`对象的`responseText`属性中找到。所以可以编写类似下面的代码：

```
request.onload = function() {
        if (request.status == 200) {
            alert(request.responseText);
        }
};
```

请求接收到一个响应时，会调用这个函数。

可以从request对象的responseText属性得到响应。

不过先等等，就要到关键了，下面将使用`request.responseText`编写一些真正的代码。

代码磁贴

*http://wickedlysmart.com/ifeelluckytoday*提供了一个新的Web服务，点击时可能返回“unlucky”也可能返回“lucky”。具体的逻辑基于一个古老的秘密算法，这里我们不便批露，不过，这是一个很不错的服务，可以让用户知道某一天他们是否幸运。

我们需要你的帮助来创建一个参考实现，向别人展示如何把这个服务包含到他们的网站中。下面给出了骨架代码，请帮我们用磁贴填入具体代码。要当心，并不是所有磁贴都会用到。我们已经帮你完成了一个空。

```
window.onload = function () {

    var url = "http://wickedlysmart.com/ifeelluckytoday";
    var request = ____________________

    _________________________ {
        if (___________________) {
            displayLuck(__________________);
        }
    };

    ___________________________
    ______________________
}
function displayLuck(luck) {
    var p = document.__________________("luck");
    p.__innerHTML__ = "Today you are " + luck;
}
```

把磁贴放在这里！

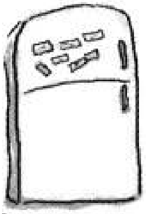


磁贴：

- `new TextHttpRequest();`
- `request.create("GET", url);`
- `var i = 0;`
- `request.responseText`
- `request.send(null);`
- `request.open("GET",url);`
- `request.onload = function()`
- `myLuckyText`
- `new XMLHttpRequest();`
- `request.status == 200`
- `getElementById`

pin51.com 拼吾爱 最新资源分享

pin51.com 拼吾爱



代码磁贴答案

*http://wickedlysmart.com/ifeelluckytoday*提供了一个新的Web服务，点击时可能返回“unlucky”也可能返回“lucky”。具体的逻辑基于一个古老的秘密算法，这里我们不便批露，不过，这是一个很不错的服务，可以让用户知道某一天他们是否幸运。

我们需要你的帮助来创建一个参考实现，向别人展示如何把这个服务包含到他们的网站中。下面给出了骨架代码，请帮我们用磁贴填入具体代码。要当心，并不是所有磁贴都会用到。以下是我们的答案：

```
window.onload = function () {

    var url = "http://wickedlysmart.com/ifeelluckytoday";

    var request = new XMLHttpRequest();
    request.onload = function() {
        if (request.status == 200) {
            displayLuck(request.responseText);
        }
    };
    request.open("GET",url);
    request.send(null);
}

function displayLuck(luck) {
    var p = document.getElementById("luck");
    p.innerHTML = "Today you are " + luck;
}
```

把磁贴放在这里！



多余的磁贴。

```
var i = 0;
request.create("GET", url);
myLuckyText
new TextHttpRequest();
```



XMLHttpRequest闪亮登场

本周访谈：

HTTP请求对象的告白

Head First: 欢迎你，XMLHttpRequest，很高兴你能抽出百忙应邀到访。能不能跟我们说说，你在构建Web应用中起什么作用。

XMLHttpRequest: 是我开启了这个新浪潮，把外部的数据结合到Web页面中。听说过Google Maps吗？GMail呢?那都是我的功劳。实际上，如果没有我，那些根本不可能。

Head First: 怎么这么说？

XMLHttpRequest: 在我来之前，人们都在服务器端构建Web页面，创建时所有一切都要"烘制"到页面中。有了我，你完全可以出去走走，等页面构建好了再来获取数据。想想Google Maps吧：每次你调整地图上的位置时，它就会更新页面显示，根本不用重新加载整个页面。

Head First: 这么说，你很成功喽！你的秘诀是什么？

XMLHttpRequest: 低调低调，嗯，很简单。给我一个URL，我就会为你获取数据。除了这个，我没有别的要求了。

Head First: 就这么简单？

XMLHttpRequest: 嗯，你确实得告诉我，等我拿到数据之后要怎么处理。你可以给我一个处理函数，算是一种回调吧。我拿到数据时，就会把数据交给你的处理程序，按你希望的方式处理数据。

Head First: 你说的是什么类型的数据？

XMLHttpRequest: 如今，Web充满了各种各样的数据：天气数据、地图、人们的社交数据、附近的地理定位数据…… 真的，你能想到的任何数据集都可以采用一种适合我的方式溶入Web。

Head First: 这都是XML数据，对吧？我的意思是，你的名字最前面有"XML"。

XMLHttpRequest: 是这样吗？你是专业人士，这就是你做这个访谈的目的吧？你肯定提前做了功课，你想说的是"你就只知道XML，是吧？"让我来纠正一下。没错，有一段时间我主要获取XML，不过这个世界日新月异。如今，我可以获取各种各样的数据。当然，有些是XML，但更多的是JSON请求。

Head First: 是吗？什么是JSON，为什么它这么流行？

XMLHttpRequest: JSON是JavaScript对象记法（JavaScript Object Notation），它有很多好处，规模、可读性等，另外还有一点：它是Web上最流行的编程语言（当然，就是我的朋友JavaScript）的内置记法。

Head First: 不过，不是说格式对你来说没有影响吗？用户应该能请求XML或JSON，或者甚至是电传通信请求，这些对你来说都无所谓，难道不是吗？

XMLHttpRequest: <不作声>

Head First: 嗯，看来戳到痛处了。没关系，正好我们也该告一段落了…… 那么，XMLHttpRequest，我想这一章后面我们再找个时间聊聊？

XMLHttpRequest: 行吧，遗憾的是，我要看看我的日程安排……

pin51.com 拼吾爱

XML让位，JSON登场

你可能记得（也可能不记得），曾几何时，我们都把XML当作救星，这是一种人类可读、机器可解析的数据格式，也是原本要支持世界上所有数据需求的数据格式。刚开始开发XMLHttpRequest时，XML确实是大家交换数据采用的方法（XMLHttpRequest也正是因此而得名）。

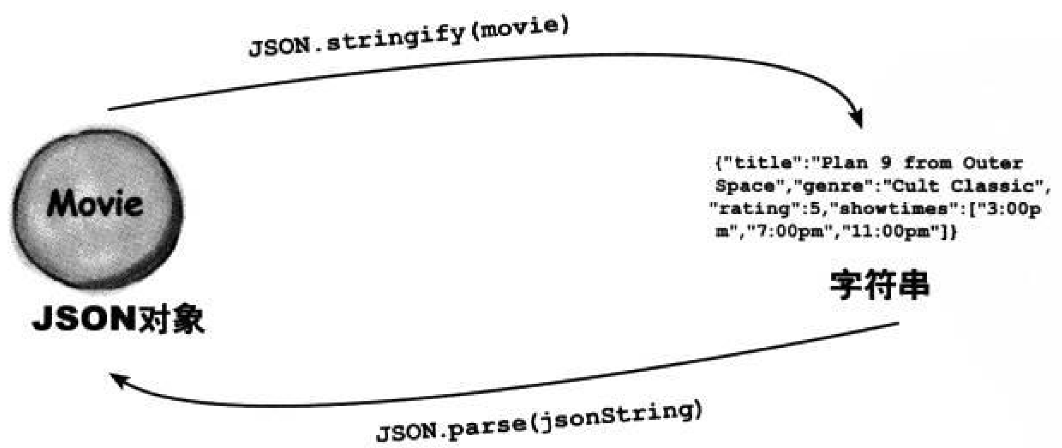
不过，在后来的道路上，显然XML踩到了JSON扔出的香蕉皮。JSON是什么？这是最新最棒的数据格式，脱胎于JavaScript，已经在整个Web得到采用，包括浏览器和服务器端。多说一句，它会不会很快成为HTML5应用选择的格式？

那么，JSON有什么过人之处呢？是这样的，它的可读性相当好，而且可以很快、很容易地解析为JavaScript值和对象。与XML不同，它非常可爱……说了这么多，你对它是不是有点点喜欢了？这本书里会大量看到JSON。我们用它在网络上交换JavaScript数据，用Web存储API将数据存储到一个本地存储库，它还是另一种访问Web数据的方法的重要部分（稍后再做更多讨论）。

不行，先等等，网络数据交换格式…… 存储格式…… 这很复杂吧，是吗？不用担心，~~接下来10页我们就会让你成为一个专家~~，实际上关于JSON需要了解的一切你都已经知道了。要想使用JSON，你只需要了解JavaScript对象（你肯定已经非常了解了）和两个简单的方法调用。它的工作是这样的：

❶ 我们有一个想交换或存储的JavaScript对象，所以调用JSON.stringify方法，把这个对象作为实参传入。

❷ 结果是表示这个对象的一个字符串。可以保存这个串，把它传给一个函数，或者通过网络发送等。



❹ 结果是原始对象的一个副本。

❸ 要把这个字符串转换回一个对象，需要把它传给JSON.parse方法。

pin51.com 拼吾爱

使用JSON的一个小例子

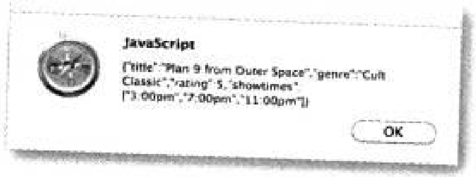
① 下面运行一个简短的小例子，将一个对象转换为它的JSON串格式。可以从你已经了解的一个对象开始：第4章中的Movie对象。并不是所有一切都能转换为一个JSON串。例如，方法就不能转换为JSON串—不过所有基本类型，比如数字、字符串和数组都可以转换。下面创建一个对象，然后把它转换为一个串：

这里确实还有另外一些限制，不过我们先不用担心这些限制。

```
var plan9Movie = new Movie("Plan 9 from Outer Space","Cult Classic", 2,
                           ["3:00pm", "7:00pm", "11:00pm"]);
```

这是一个完整的movie对象，包括字符串、数字和一个数组。

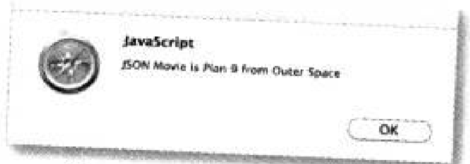
② 一旦得到对象，可以用`JSON.stringify`方法把它转换为JSON串格式。下面来看这是如何做到的(你可以试试看，打开第4章的movie代码，把下面的代码增加到脚本的最后)：




```
var jsonString = JSON.stringify(plan9Movie);
alert(jsonString);
```

这是得到的结果，提醒中会显示对象的字符串版本。

③ 我们得到了一个表示Movie对象的JSON串。现在可以对这个字符串做任何处理，比如通过HTTP把它发送到一个服务器。另外还可以从另一个服务器接收一个JSON串。假设一个服务器为我们提供了这样一个串；怎么把它转换回一个对象来进行处理呢？只需要使用`JSON.stringify`的姊妹方法：JSON.parse。如下所示：




```
var jsonMovieObject = JSON.parse(jsonString);
alert("JSON Movie is " + jsonMovieObject.title);
```

哈，现在把它作为一个真正的对象，访问它的属性。

BRAIN POWER

试试这个URL。你看到了什么？

http://search.twitter.com/search.json?q=hfhtml5

说明：Firefox会要求你打开或保存一个文件。可以用TextEdit、Notepad或者任何基本的文本编辑器打开。



拼吾爱

规范刚刚收到！

Gumball服务器规范

万能糖果公司

有了糖果机，总是充满活力

非常感谢你能挑起重担！

我们已经把糖果机发来的所有销售信息汇总，并由我们的中心服务器发布：

http://gumball.wickedlysmart.com/

我们选择JSON作为数据格式，如果你点击上面的URL，会得到一个JSON对象数组，类似下面这样：

```
[{"name":"CAMPBELL",
  "time": 1302212903099,
  "sales": 3},

 {"name": "FRESNO",
  "time": 1302212903100,
  "sales": 2},

  . . .
]
```

城市名：目前我们只测试加利福尼亚州的情况。

报告到来的时间（单位为毫秒）。

上一次报告之后售出的糖果数。

第二个城市，弗雷斯诺。

这里还可能有更多城市……

一定要做到！

在你的浏览器中键入这个URL，看看返回的值。你会看到一个数组，其中有一个或多个这样的对象。

还可以在URL的末尾增加一个*lastreporttime*参数，这样只会得到这个时间之后的报告。可以像下面这样使用：

只需指定一个时间（单位为毫秒）。

http://gumball.wickedlysmart.com/?lastreporttime=1302212903099

现在我们得到了数百个糖果机报告，实际上平均大约每5～8秒都会看到报告。也就是说，这是我们的生产服务器，所以请先在本地测试你的代码！

再次感谢你的帮助！另外要记住，就像我们的CEO所说："有了糖果机，永远充满活力"。

——万能糖果公司工程师

pin51.com 拼吾爱

开始工作吧！

我们已经拿到万能糖果公司的规范，另外也完成了XMLHttp-Request和JSON的培训。你应该已经准备好了，下面来编写一些代码，让你的第一个糖果应用运行起来。

要记住，我们已经完成了一些HTML，可以作为起点，它链接到一个名为`mightygumball.js`的文件。我们就从这个文件开始编写代码。另外还要记住，我们在HTML中留了一个位置来放置糖果销售数据，就放在id为“sales”的`<div>`中。下面把这些内容集成起来，编写一些代码。

```html
<!doctype html>
<html lang="en">
  <head>
    <title>Mighty Gumball (JSON)</title>
    <meta charset="utf-8">
    <script src="mightygumball.js"></script>
    <link rel="stylesheet" href="mightygumball.css">
  </head>
  <body>
    <h1>Mighty Gumball Sales</h1>
    <div id="sales">

    </div>
  </body>
</html>
```

编写一个onload处理函数

相信这对你来说已经再熟悉不过了，我们要编写一个onload处理程序，HTML完全加载时就会调用这个处理程序。我们还要发出一个HTTP请求来得到销售数据。数据返回时，我们会要求`XMLHttpRequest`调用函数`updateSales`（稍后就来编写这个函数）：

```javascript
window.onload = function() {
    var url = "http://localhost/sales.json";
    var request = new XMLHttpRequest();
    request.open("GET", url);
    request.onload = function() {
        if (request.status == 200) {
            updateSales(request.responseText);
        }
    };
    request.send(null);
}
```

首先测试一个本地文件（遵照万能糖果公司工程师的建议），确保一切正常。稍后会更详细地讨论这个内容……

这里创建了对象，基于我们的URL调用open方法，然后将onload属性设置为一个函数，建立XMLHttpRequest。

检查是否一切正常，然后……

……数据完成加载时，会调用这个函数。

最后，发出请求。



如果你在使用Opera、IE 8或更早版本，建议你用其他浏览器进行测试。后面会讨论如何支持Opera和更老版本的IE浏览器。

pin51.com 拼吾爱

显示糖果销售数据

现在要编写处理程序updateSales。先让问题容易一些，来看最简单的实现，以后可以让它变得更好：

```
function updateSales(responseText) {
    var salesDiv = document.getElementById("sales");
    salesDiv.innerHTML = responseText;
}
```

获取HTML中的<div>，用它放置数据。

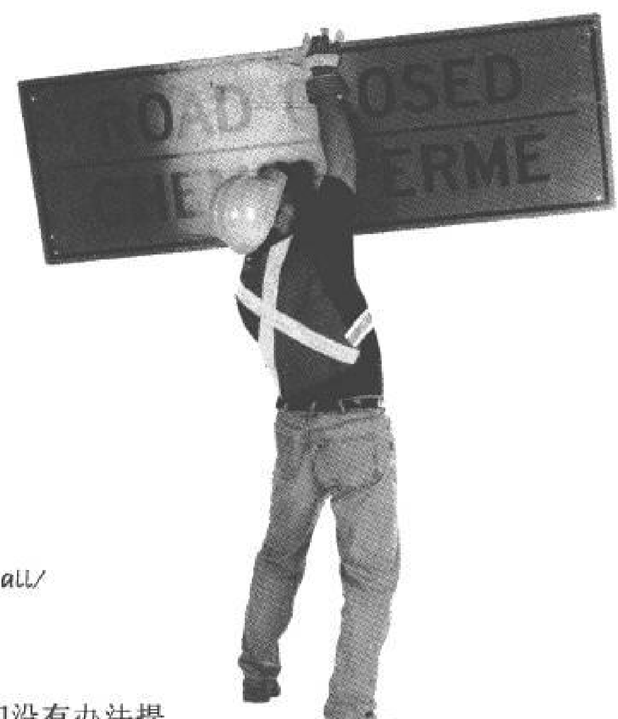
把这个div的内容设置为整个数据块。稍后会处理具体解析……先来做个测试。

当心，前面要绕行！

又该测试了，不过首先要稍稍绕点路。万能糖果公司工程师要求我们访问他们的生产服务器之前先在本地测试，这是个好主意。不过要完成本地测试，我们需要把这些数据放在一个服务器上，这样XMLHttpRequest才能使用HTTP协议获取数据。

对于服务器，可以有几个选择：

- 如果你的公司已经提供了一个可用来完成测试的服务器，可以直接使用。
- 或者，如果可以使用一个第三方托管服务，如GoDaddy、Dreamhost或另外某个托管公司。
- 最后，可以在你自己的机器上建立一个服务器。如果是这种情况，你的URL应该类似于：



```
http://localhost/mightygumball.html
```

文件也可以放在一个子目录中，如http://localhost/gumball/mightygumball.html

查看下一页的提示和线索。要记住，托管环境会稍有差别，我们没有办法提供一个面向所有托管环境的通用指南。所以，愿力量与你同在，如果你不能轻松地访问一个已有的服务器，在你的本地机器上建立一个服务器可能是你的最佳选择！

绕路

如何建立你自己的Web服务器

如何建立你的本地托管，这很大程度上取决于你使用哪种操作系统。查看以下对OS X（也称为Mac）、PC和Linux的提示。下一页还会看到另外一些选择。



我是一个Mac

在Mac上建立Web服务器很容易。进入 > System Preferences（系统首选项），然后选择Sharing（共享）。在左边的面板中，一定要选中Web Sharing（Web共享）：



一旦打开Web Sharing（或者如果已经打开），你会看到一些信息，告诉你如何访问你的本地服务器。应该能使用localhost而不是IP地址（如果你使用DHCP路由器，IP地址可能会变，所以localhost更合适）。默认地，你的文件将由`http://localhost/~YOUR_USERNAME/`提供，这会从你的`YOUR_USERNAME/Sites/`文件夹提供，所以可以在那里为万能糖果公司建立一个子文件夹。



我是一个PC

在Windows上安装你自己的Web服务器要比从前容易得多，这要归功于Microsoft Web Platform Installer（也称为Web PI）。Windows 7、Windows Vista SP2、Windows XP SP3+、Windows Server 2003 SP2+、Windows Server 2008和Windows Server 2008 R2上都提供了Web PI的当前版本，另外可以从这里下载：`http://www.microsoft.com/web/downloads/platform.aspx`。

还有一个选择，可以安装随Apache、PHP和MySQL提供的开源WampServer，用来完成Web应用开发。这个服务器很容易安装和管理。

可以从`http://www.wampserver.com/en/`下载WampServer。

如果你留心还会看到很多其他开源的解决方案，选择确实很多。

我是~~全能~~Linux发布

面对现实吧，你知道你要做什么，对不对？一般都会默认安装Apache，所以可以查看你的发布文档。

绕路

如何建立你自己的Web服务器（续）

哈，你想真正托管你的页面？太棒了，让页面真正在Web上托管，这绝对是无可替代的。参考下面的提示，祝你开心！

第三方托管……

如果你不想建立你自己的服务器，完全可以使用一个远程服务器，不过需要托管你的HTML、JavaScript和CSS，还有JSON文件，所有这些都要放在同一个服务器上（后面会讨论为什么这一点至关重要），这样才能顺利完成这个例子。

大多数托管服务允许你通过FTP访问一个文件夹，你可以把所有这些文件都放在这个文件夹里。如果你能访问这样一个服务器，可以上传所有文件，把下一页上的所有localhost换成你的服务器名。



如果不想使用命令行FTP，完全可以使用一个FTP程序（如Transit、Cyberduck或WinSCP）来得到你上传的文件。

也许你需要我们推荐托管服务，所以我们为你整理了一组托管提供商，不过托管服务商很容易找到，只需要搜索“web托管”，你就会找到大量托管服务可供选择。我们的列表放在`http://wickedlysmart.com/hfhtml5/hosting/hosting.html`。如果你上线了一个HTML5网站，请通知我们，我们会非常期待！

pin51.com 拼吾爱

再来看看代码

希望你已经建立了自己的服务器，而且已经顺利运行。这可以是你的本机上运行的一个服务器（我们就是这样做的），也可以是你能访问的其他位置上的一个服务器。不论哪种情况，总之要把你的HTML和JavaScript文件放在这个服务器上，然后将浏览器指向HTML文件。还需要把万能糖果公司销售数据测试文件也放在那里，所以我们将提供一个简单的数据文件，可以把它放在你的服务器上。对于你的应用来说，看起来数据就像是从万能糖果公司的近实时服务器生成的，这样就允许你测试代码而无需访问万能糖果公司的生产服务器。这个文件名为sales.json，随本书代码提供（或者如果你喜欢打字，也可以自己输入），内容如下：

```
[{"name":"ARTESIA","time":1308774240669,"sales":8},
 {"name":"LOS ANGELES","time":1308774240669,"sales":2},
 {"name":"PASADENA","time":1308774240669,"sales":8},
 {"name":"STOCKTON","time":1308774240669,"sales":2},
 {"name":"FRESNO","time":1308774240669,"sales":2},
 {"name":"SPRING VALLEY","time":1308774240669,"sales":9},
 {"name":"ELVERTA","time":1308774240669,"sales":5},
 {"name":"SACRAMENTO","time":1308774240669,"sales":7},
 {"name":"SAN MATEO","time":1308774240669,"sales":1}]
```

访问真正的生产服务器得到实时销售数据之前，我们先用"sales.json"进行测试。

把这个文件放在你的服务器上，并确保将JavaScript更新为这个文件的URL。我们的URL是：

http://localhost/gumball/sales.json:

先在浏览器中测试这个URL，确保一切正常，这很有帮助。




```
window.onload = function() {
    var url = "http://localhost/gumball/sales.json";
    var request = new XMLHttpRequest();
    request.open("GET", url);
    request.onload = function() {
        if (request.status == 200) {
            updateSales(request.responseText);
        }
    };
    request.send(null);
}
```

确保这指向正确的URL。

pin51.com 拼吾爱

来测试吧！

真是一条漫长的路，不过终于要测试这个代码了！

确保已经把HTML、JavaScript、JSON，还有别忘了你的CSS文件都放在服务器上。把HTML文件的URL输入到浏览器（我们的URL是`http://localhost/gumball/mightygumball.html`），再按回车键……



虽然不够美观，不过确实有数据了。

记住，我们在发送一个HTTP请求来得到sales.json中的数据，这些数据刚刚放在<div>中。看上去一切顺利！

如果遇到问题，可以通过你的浏览器单独查看各个文件，确保每个文件都可以访问。然后反复检查你的URL。



pin51.com 拼吾爱 最新最全资源分享下载站

让客户震撼……

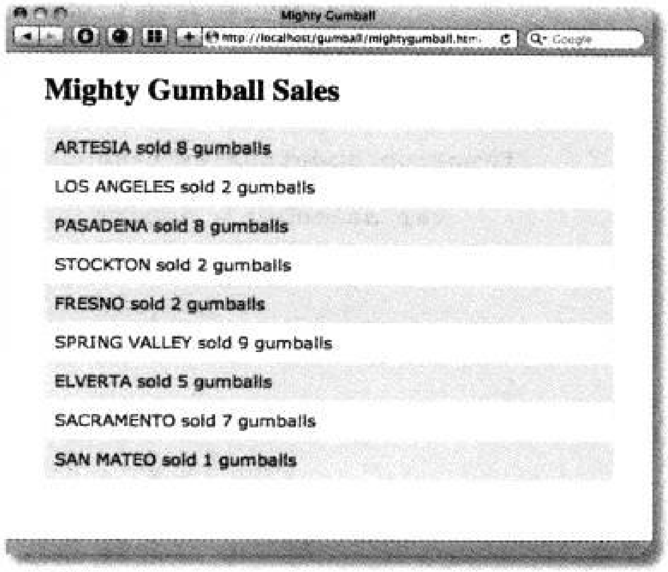
经过这么多艰苦努力，这个应用已经能运转了，很不错，不过万能糖果公司还希望更上一层楼，如果能看上去漂亮些，他们会更为震撼。现在就来做这个工作……

我们现在的应用



目前我们只是把一个JSON数组直接放在浏览器中。尽管可行，不过很丑陋。而且真是浪费，有这样一个很好的数据结构居然没能更有效地利用！

我们想要的应用



这里我们充分利用了JSON数组，由它创建了一个漂亮的显示。就是这最后的10%区分出了什么是专业，什么是业余，你觉得呢？

需要从这些方面改进显示：

1. 首先，需要从XMLHttpRequest对象取回数据（这是一个JSON串），把它转换为一个真正的JavaScript对象。

2. 然后遍历得到的数组，向DOM增加新元素，对应数组中的每个销售数据增加一个元素。

pin51.com 拼吾爱

调整代码以利用JSON

下面就按照这两个步骤调整代码：

① 首先需要从XMLttpRequest对象取回数据（这是一个JSON串），把它转换为一个真正的JavaScript对象。

为了做到这一点，下面更新updateSales函数，首先删除将<div>内容设置为responseText串的那一行代码，使用JSON.parse把responseText从字符串转换为相应的JavaScript对象。

```
function updateSales(responseText) {

    var salesDiv = document.getElementById("sales");

    salesDiv.innerHTML = responseText;

    var sales = JSON.parse(responseText);

}
```

~~salesDiv.innerHTML = responseText;~~ ← 不再需要这行代码了。

得到响应，使用JSON.parse把它转换为一个JavaScript对象（在这里，这将是一个数组），把它赋给变量sales。

② 现在遍历所得到的数组，向DOM增加新元素，对于数组中的每个销售数据，在DOM中增加一个元素。我们将为每个销售数据创建一个新的<div>：

```
function updateSales(responseText) {
    var salesDiv = document.getElementById("sales");
    var sales = JSON.parse(responseText);
    for (var i = 0; i < sales.length; i++) {
        var sale = sales[i];
        var div = document.createElement("div");
        div.setAttribute("class", "saleItem");
        div.innerHTML = sale.name + " sold " + sale.sales + " gumballs";
        salesDiv.appendChild(div);
    }
}
```

迭代处理数组中的每一项。

对于数组中的每一项创建一个<div>，并指定类（class）为"saleItem"（CSS会用到）。

用innerHTML设置<div>的内容，然后增加这个元素，作为sales <div>的子元素。

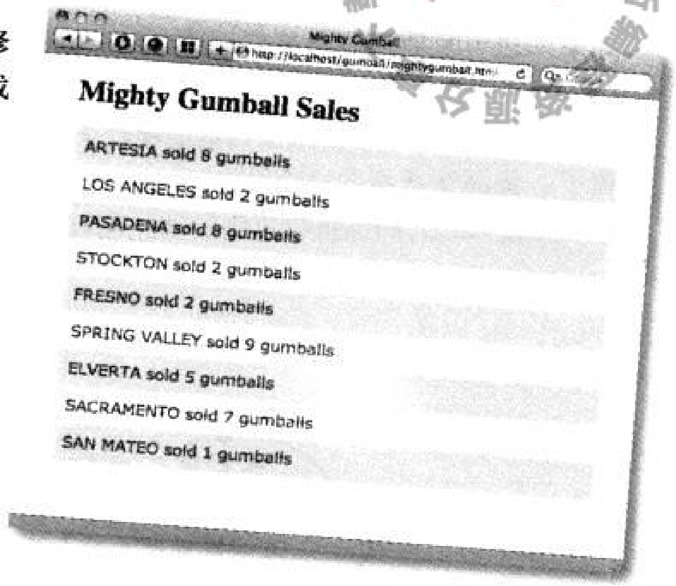
pin51.com 拼吾爱

最后冲刺……



你已经知道这个应用最后是什么样子，不过还需要继续做以下修改。再仔细看看上一页的代码，确保已经全部掌握。然后重新加载这个页面。

看吧，我们说过最后会是这个样子！



测试一切正常，现在你们可以用万能糖果公司的实际生产服务器了。祝你好运！



Ajay，质量保证检验员。

转向实际服务器

万能糖果公司要求我们先完成本地测试，我们做到了。现在可以转向在真正的服务器上完成测试。这一次，我们不再获取一个静态的JSON数据文件，而是要获取从万能糖果公司服务器动态生成的JSON。需要更新XMLHttpRequest使用的的URL，把它改为指向万能糖果公司的URL。下面来做这个工作：

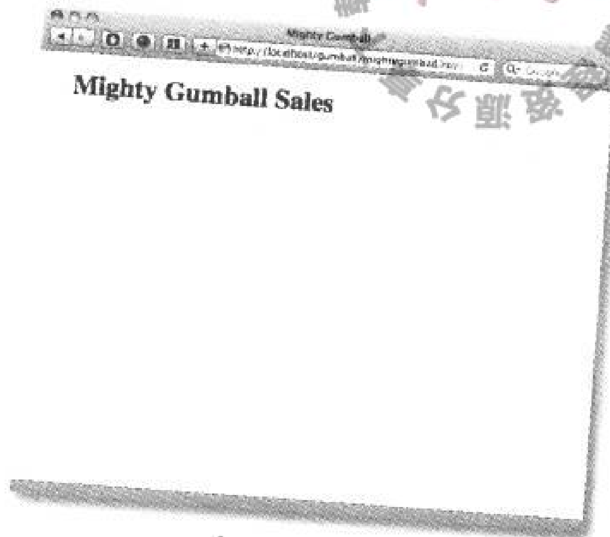
这是他们的服务器URL。改变这个URL，一定要保存。

```
window.onload = function() {
    var url = "http://gumball.wickedlysmart.com";
    var request = new XMLHttpRequest();
    request.open("GET", url);
    request.onload = function() {
        if (request.status == 200) {
            updateSales(request.responseText);
        }
    };
    request.send(null);
}
```

pin51.com 拼吾爱

现场测试……

如果你想不断地从服务器获取HTML，就要把所做的URL修改保存到服务器上的mightygumball.js文件中，否则，如果你在使用localhost，则要保存到本地硬盘上。现在你应该知道怎么做了：将浏览器指向这个HTML文件，你会看到从世界各地购买万能糖果的人们那里收集到的漂亮而真实的现场数据！



怎么了？什么数据也看不到！

休斯顿，我们遇到一个问题。快来看看，自从我们换成实际的服务器之后，就再没得到过销售数据！



唉呀！

看起来一切都很顺利；我们原本以为现在可以喝着饮料庆祝又为万能糖果公司完成了一个成功的项目。可是情况突然急转直下。是的，有点太戏剧性了，不过到底怎么回事？应该能正常工作的！

致编辑：我们原本以为已经提前做了充分的检查，完全可以拿去印刷了！不过现在还得再来解决这个小问题！

放松，深呼吸。来吧，做个合理的解释……

Ajay，部门的质量保证检验员。

www.1n5i.com 拼吾爱 最新编程



惊险情节！

页面上没有看到任何数据。在转向实际服务器之前本来一切都很正常的……

能找出问题吗？

能修正吗？

别走开…… 我们会回答这些问题，甚至不只这些问题……

与此同时，看看你有什么想法：哪里出了问题？如何来修正？

BULLET POINTS

- 要从服务器得到HTML文件或数据，浏览器会发出一个HTTP请求。
- HTTP响应包括一个响应码，指示请求是否有错误。
- HTTP响应码200表示请求没有错误。
- 要从JavaScript发出一个HTTP请求，需要使用XMLHttpRequest对象。
- XMLHttpRequest对象的onload处理程序处理从服务器获取响应。
- XMLHttpRequest的JSON响应放在请求的responseText属性中。
- 要把responseText串转换为JSON，可以使用JSON.parse方法。
- 应用中可以使用XMLHttpRequest更新内容，如地图和email，而无需重新加载页面。
- XMLHttpRequest可以用来获取任何类型的文本内容，如XML、JSON等。
- XMLHttpRequest Level 2是XMLHttpRequest的最新版本，不过相关标准仍在开发中。
- 要使用XMLHttpRequest，必须从一个服务器提供文件和请求数据。可以在你自己的机器上建立一个本地服务器进行测试，也可以使用一个托管解决方案。
- 一些较老的浏览器不支持XMLHttpRequest onload属性，如IE8和更低版本，以及Opera 10和更低版本。可以编写代码来检查浏览器版本，为较老的浏览器提供一个替代方案。



XMLHttpRequest闪亮登场（第2部分）

本周访谈：
Internet Explorer，“你说过JSON吗？”

Head First: 欢迎回来，继续我们访谈的第2部分。我想问问你有关浏览器支持的问题，是不是只有新的浏览器才支持你呀？

XMLHttpRequest: 人们不会无缘无故叫我“前辈”，从2004年以来我就得到了浏览器的支持。在互联网时代，我要算老资格了。

Head First: 嗯，那会不会过时呢，你担心过吗？

XMLHttpRequest:我可是个善于自我改进的人，每过十年我就会以新的面貌出现。现在，我们使用的是XMLHttpRequest的第2个版本，称为Level 2。实际上，大多数现代浏览器已经支持Level 2。

Head First: 真不错。Level 2又有什么不同呢？

XMLHttpRequest: 嗯，首先，它支持更多事件类型，这样你就能做更多的事情，比如跟踪一个请求的进度，还可以编写更精巧的代码（在我看来）。

Head First: 说到浏览器支持……

XMLHttpRequest: 哈，终于来了……一直等着它呢……

Head First: 我们听到一些谣言，说你和IE的关系不怎么样……

XMLHttpRequest: ……确实有这样一些谣传……如果你希望我回答这个问题，仔细看看之前对我的访谈就可以了。不过，显然你没有好好研究过。你是开玩笑吗？XMLHttpRequest的一切正是从IE开始的。

Head First: 是啊，不过ActiveXObject和XDomainRequest呢？你以前听说过这些名字吗？

XMLHttpRequest: 这是我的外号呀！在Microsoft他们就是这么叫我的！没错，我承认，我有不同的名字是有些麻烦，不过它们做的事情都一样。只需要多写一点点代码就能很容易地处理，而且对于最新的Microsoft浏览器来说，也就是版本9和以后的版本，完全没有问题。如果读者还不太了解这方面，我很乐意在访谈之后留下来，确保他们的代码在较老版本的IE上也能正常工作。

Head First: 你心肠真好，我们相信这一章肯定会有这方面的内容。

XMLHttpRequest: 嘿，我可是个好人，我不会抛下读者不管的。

Head First: 我们相信你。不过还有一个问题：你提到JSON，你是它的狂热粉丝。你担心过JSONP吗？

XMLHttpRequest: 什么？我？担心？

Head First: 大家都在传言，说很多人都在使用JSON来取代你。

XMLHttpRequest: 对，当然，利用JSONP确实可以获取数据，不过这只是个聪明的手段。我的意思是，想想你要写的那些复杂的代码，另外再想想安全性。

Head First: 拜托，我可不那么专业，我只知道很多人说它能解决你没办法解决的问题。不管怎样，时间到了，我们的访谈只能先告一段落了。

XMLHttpRequest: 哈，真遗憾，不过至少你说“不那么专业”，这确实是事实。

pin5i.com 拼吾爱

Watch it!

较老版本的浏览器不支持XMLHttpRequest onload属性，不过有一种简单的方法可以绕过这个问题。

我们一直在使用request.onload定义一个函数，从服务器获取数据的请求完成时就会调用这个函数。这是XMLHttpRequest Level 2（可以认为它是“版本2”）的一个特性。XMLHttpRequestLevel2还相当新，所以很多用户可能还在使用不支持这个特性的浏览器。具体来讲，IE 8（及更低版本）和Opera 10（及更低版本）就只支持XMLHttpRequest Level 1。有一个好消息，XMLHttpRequest Level 2的新特性都作为改进实现，所以你还可以继续在所有浏览器中使用版本1的全部特性，不会有任何问题；只是这样一来你的代码可能不能算精巧。下面是使用XMLHttpRequest Level 1的代码：

使用XMLHttpRequest Level 1的大多数代码都是一样的……

```
function init() {
    var url = "http://localhost/gumball/sales.json";
    var request = new XMLHttpRequest();
    request.onreadystatechange = function() {
        if (request.readyState == 4 && request.status == 200) {
            updateSales(request.responseText);
        }
    };
    request.open("GET", url);
    request.send(null);
}
```

……不过，Level 1中没有request.onload属性，所以应当使用onreadystatechange属性。

然后检查readyState，确保已经完成数据加载。如果readyState为4，可以知道数据已经加载。

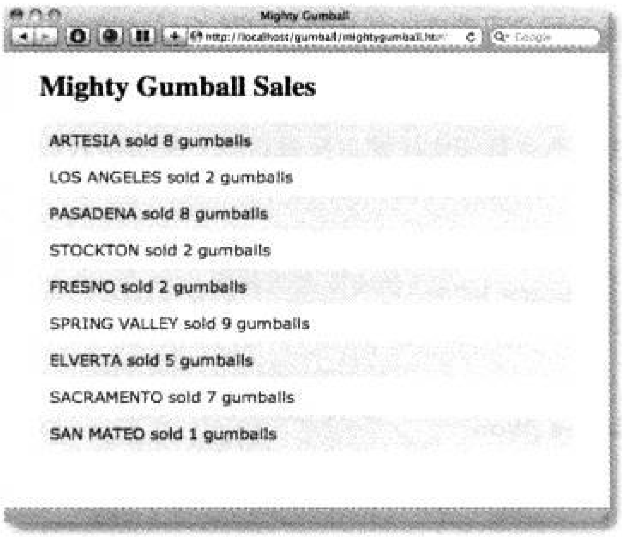
如果希望检查其他错误，还可以检查其他readyState和status值。

其他代码基本上都是一样的。

记得吧？我们有一个惊险情节，这是一个bug

用我们的本地服务器时，所有代码都能运行得很好，不过，转向Web上的实际服务器时，却失败了！

我们期望的：



这是使用我们的本地服务器从`http://localhost/gumball/sales.json`提供销售数据时运行代码得到的页面。

我们得到的：



这是使用实际万能糖果公司服务器从`http://gumball.wickedlysmart.com`提供销售数据时运行代码看到的页面。

那么，现在该怎么办？！

没什么，平常怎么做就怎么做，把相关人员召集起来开个小会。相信经过共同努力，我们（包括一些虚构的人物）一定能找出问题在哪里！Frank? Jim? Joe? 你们都在哪儿？噢，原来你们已经到下一页了……



Ajay，质量保证检验员，相当郁闷。

pin5i.com 拼吾爱 最新资源分享下载站



Jim: 你的URL写对了吗？

Frank: 对呀，实际上，我还把它键入到浏览器，确保能看到期望的销售数据，一切都正常啊。真搞不懂……

Joe: 我瞥了一眼Chrome上的JavaScript控制台，看到一些与访问控制、来源或域有关的东东。





pin51.com 拼吾爱

浏览器安全策略是什么？



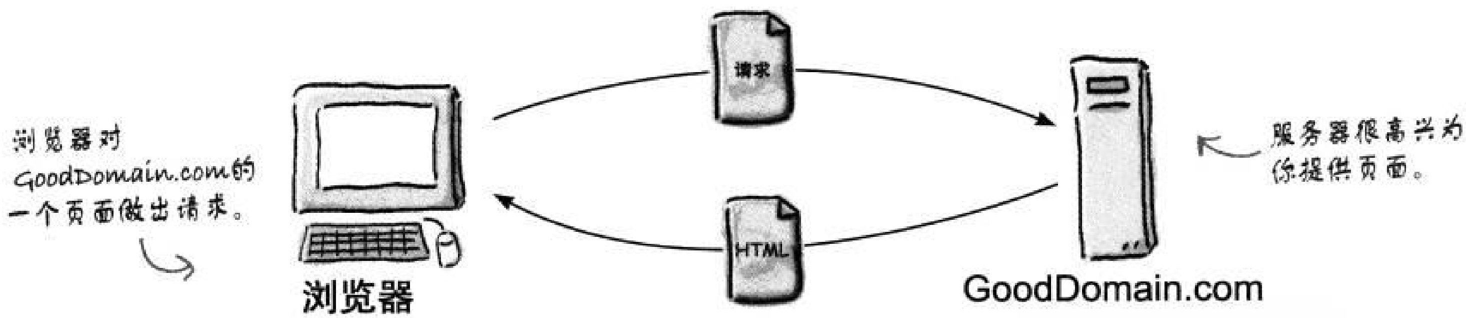
唉，碰到这种障碍真让人难堪，设身处地为读者们想想吧。不过，Judy说得对，浏览器确实对XMLHttpRequest HTTP请求有一些安全限制，这可能会导致一些问题。

那么，这个策略是什么？嗯，这是一个浏览器策略，如果从某个域提供页面本身，安全策略要求不能从另一个域获取数据。想象一下，假设你在运行DaddyWarBucksBank.com网站，有人闯入了你的系统，插入一些JavaScript，拿走了用户的个人信息，并与服务器HackersNeedMoreMoney.com通信对这个信息为所欲为。听起来不太好吧，对不对？嗯，为了防止诸如此类的事情发生，浏览器不允许你对原先提供页面的域以外的其他域发出XMLHttpRequests请求。

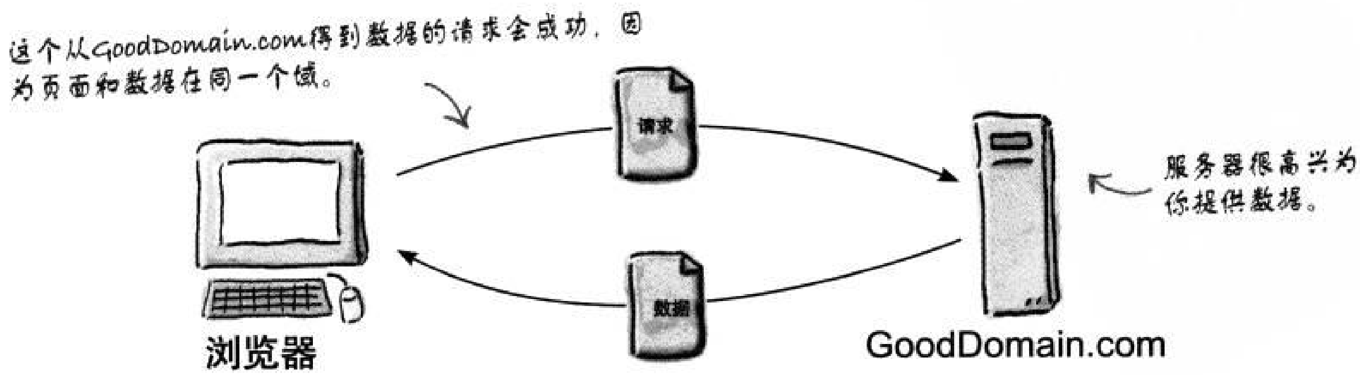
下面就来看看，哪些是允许的，而哪些不可以：

JavaScript代码可接受的行为：

1. 首先，用户（通过浏览器）对一个HTML页面做出请求（当然，也包括所有相关的JavaScript和CSS）：



2. 页面需要得到GoodDomain.com的一些数据，所以对这些数据做出XMLHttpRequest请求：

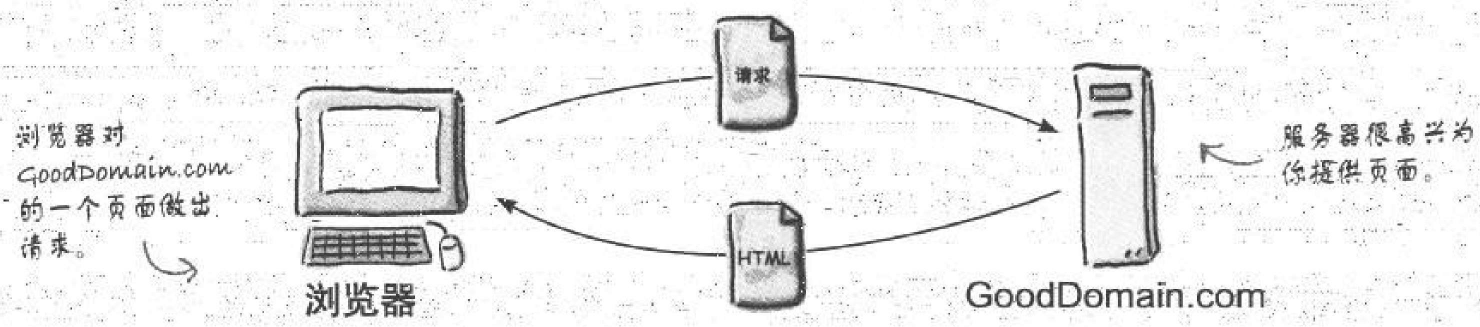


pin51.com 拼吾爱 最新最全最优秀的资源下载站

JavaScript代码不能接受的行为：

假设页面在GoodDomain.com托管，现在来看试图使用XMLHttpRequest向BadDomain.com做出数据请求时会发生什么。

1. 与前面一样，浏览器对GoodDomain.com上的一个页面做出请求，这可能包括JavaScript和CSS文件（同样位于GoodDomain.com）。



2. 不过，现在代码希望从另一个来源得到数据，也就是BadDomain.com。下面来看页面使用XMLHttpRequest请求这个数据时会发生什么：



pin51.com 拼吾爱 最新最全资源分享下载站

嘿，所有这些代码居然不能工作？难道不能把我们的文件直接复制到万能糖果服务器吗？



通常答案是肯定的。

假设你是一个开发人员，在为万能糖果公司开发代码，通常你能访问他们的服务器（或者可以与能够帮你在服务器上部署文件的人联系），你可以把你的所有文件都放在那里，这就能避免跨域问题了。不过，在这里不行（也许你已经沉浸在你的遐想中，我们真的不愿意打断你），事实上你并没有为万能糖果公司工作，你只是这本书的读者，我们没办法让几十万人都把他们的文件复制到万能糖果服务器。

至少光靠编辑给我们的预算，这是办不到的！

这会把我们置于何地？难道说我们进了死胡同，再没有指望了？不，我们还有很多选择。下面就来逐个了解……



那么，我们有哪些选择？

说实话，我们很清楚XMLHttpRequest跨域请求会失败。不过，正像我们所说的，构建应用时你能够访问服务器，所以这不会成为问题（如果所构建的应用主要依赖你自己的数据，使用XMLHttpRequest通常是最好的办法）。

但是，此时此刻，你可能会说“那很好，不过我们怎么让这个代码正常工作呢？”嗯，要让它正常工作有两种方法：

1. **计划1：使用我们的托管文件。**

   我们已经在我们的服务器上为你存放了文件，位于：

   `http://gumball.wickedlysmart.com/gumball/gumball.html`

   可以试一试，让浏览器指向这个URL，你应该能看到目前键入的代码居然起作用了。

2. **计划2：用其他办法获得数据。**

   这么说，在应用所在的同一个域托管数据时，XMLHttpRequest是为应用获取数据的一种很好的方法，不过，如果需要从第三方获取数据呢？例如，假设你需要Google或Twitter的数据呢？在这些情况下，我们确实必须换个角度看问题，需要另辟蹊径。

   确实还有一种基于JSON的方法，称为JSONP（如果你想知道它代表什么，可以告诉你，这表示“JSON with Padding”。没错，我们也认为这听上去有些怪异，不过稍后就会向你解释）。带好你的宇航背包，因为它的工作方式有些像“来自另一个星球”（希望你明白我们的意思）。

pin51.com 拼吾爱 最新最全资源分享下载站



Joe: 太好了！不过，这到底是什么？

Jim: 听上去还有其他办法可以从Web服务为应用获取数据。

Frank: 我在这方面一点想法都没有。我只是有点创造性。

Jim: Frank，我可不认为这是坏事。我在google上简单查了一个JSONP，基本说来，这种方法就是让<script>标记完成获取数据的工作。

Joe: 呵，这样合适吗？

Jim: 当然合适—很多大型服务都支持JSONP，比如Twitter。

Frank: 听起来不错呢。

Joe: 嗯，这正是我想搞清楚的。我的意思是说，怎么能使用`<script>`标记作为一种合适的获取数据的方法呢？我甚至不明白它的做法。

Jim: 我自己也只是略知一二。不过可以这样想：使用一个`<script>`元素时，它会为你获取代码，对吧？

Joe: 对呀……

Jim: 嗯，那如果你把数据放在代码中呢？

Joe: 噢，有点开窍了，轮子转起来了……

Frank: 哈，你是说仓鼠玩的轮子吧……

pin51.com 拼吾爱





HTML5高手：……现在也是如此。菜鸟，看看这个代码：

```
alert("woof");
```

它要做什么？

Web开发人员：执行这个代码时，假设在一个浏览器中运行，它会显示一个提醒，指示“woof”。

高手： 哈，没错。创建你自己的一个简单的HTML文件，其中放入一个<script>元素（放在体里），像下面这样：

这个代码位于这个URL。

```
<script src="http://wickedlysmart.com/hfhtml5/chapter6/dog.js">
</script>
```

高手： 它会做什么？

Web开发人员：它会加载页面，这会从wickedlysmart.com的dog.js加载JavaScript，这个代码会调用alert函数，所以我会看到浏览器显示一个指示“woof”的提醒。

高手：这么说，由另一个域提供的JavaScript文件可以调用你的浏览器的一个函数？

Web开发人员：嗯，既然你这么说，高手，我想是这样的。dog.js文件在wickedlysmart.com，一旦获取这个文件，就会在我的浏览器中调用alert。

高手： 你会在http://wickedlysmart.com/hfhtml5/chapter5/dog2.js看到另一个文件，其中包含下面的JavaScript：

```
animalSays("dog", "woof");
```

高手：这又会做什么？

pin51.com 拼吾爱 最新最全下载站

Web开发人员： 这和dog.js很类似，不过它调用一个函数animalSays。另外它有两个参数而不是一个：动物类型和动物声音。

高手： 你来编写这个函数animalSays，并把它增加到HTML文件首部的一个<script>元素中，放在指向wickedlysmart的<script>元素上面。

Web开发人员： 这样怎么样？

```
function animalSays(type, sound) {
    alert(type + " says " + sound);
}
```

高手： 很好，你进步很快。现在，修改另一个<script>引用，把指向dog.js的引用改为指向dog2.js，再在浏览器中重新加载页面。

Web开发人员： 我得到一个提醒，告诉我"dog says woof"。

高手： 再来看看http://wickedlysmart.com/hfhtml6/chapter5/cat2.js，把你的<script>引用改为指向cat2.js，再试试。

```
animalSays("cat", "meow");
```

Web开发人员： 我得到一个提醒，告诉我"cat says meow"。

高手： 所以呀，从另一个域提供的JavaScript文件不仅可以在代码中调用它想要的任何函数，还可以为我们传递它希望的任何数据，对不对？

Web开发人员： 我还没看到数据呢，只有两个参数。

高手： 参数不就是数据吗？如果我们把参数改成这样：

```
var animal = {"type": "cat", "sound": "meow"};
animalSays(animal);
```

← cat3.js

Web开发人员： 现在为函数animalSays传入了一个参数，它刚好是一个对象。嗯，我看出来了，对象有些像数据。

高手： 你能重写animalSays，让它使用这个新对象吗？

Web开发人员： 我来试试……

pin51.com 拼吾爱

Web开发人员： 这样怎么样？

```
function animalSays(animal) {
   alert(animal.type + " says " + animal.sound);
}
```

高手： 真不错。把引用改为指向http://wickedlysmart.com/hfhtml5/chapter6/dog3.js，再来试试。另外还可以试试http://wickedlysmart.com/hfhtml5/chapter6/cat3.js。

Web开发人员： 太棒了，用我的新函数，这两个引用都能得到你期望的结果。

高手： 如果把函数名animalSays改成updateSales呢？

Web开发人员： 高手，我实在看不出来动物和糖果销售有什么关系？

高手： 先听我的吧。如果把dog3.js重命名为sales.js，再像下面这样重写代码：

```
var sales = [{"name":"ARTESIA","time":1308774240669,"sales":8},
             {"name":"LOS ANGELES","time":1308774240669,"sales":2}];

updateSales(sales);
```

Web开发人员： 我好像有点明白了。我们在通过所引用的JavaScript文件传递数据，而不是使用XMLHttpRequest自己来获取。

高手： 对的，菜鸟。不过不要只见树木，不见森林。我们是不是在从另一个域获取数据？这可是XMLHttpRequest不允许的。

Web开发人员： 对，看起来是这样。真的就像是有魔法一样。

高手： 并没有什么魔法，<script>元素就是这样工作的。这都是你一直以来的努力。现在开始坐禅，想想这是怎么做的，把它牢牢记住。

Web开发人员： 好的，师父。“牢牢记住”……这话听着很熟悉，不过我实在想不起来在哪里听过。



坐禅时间

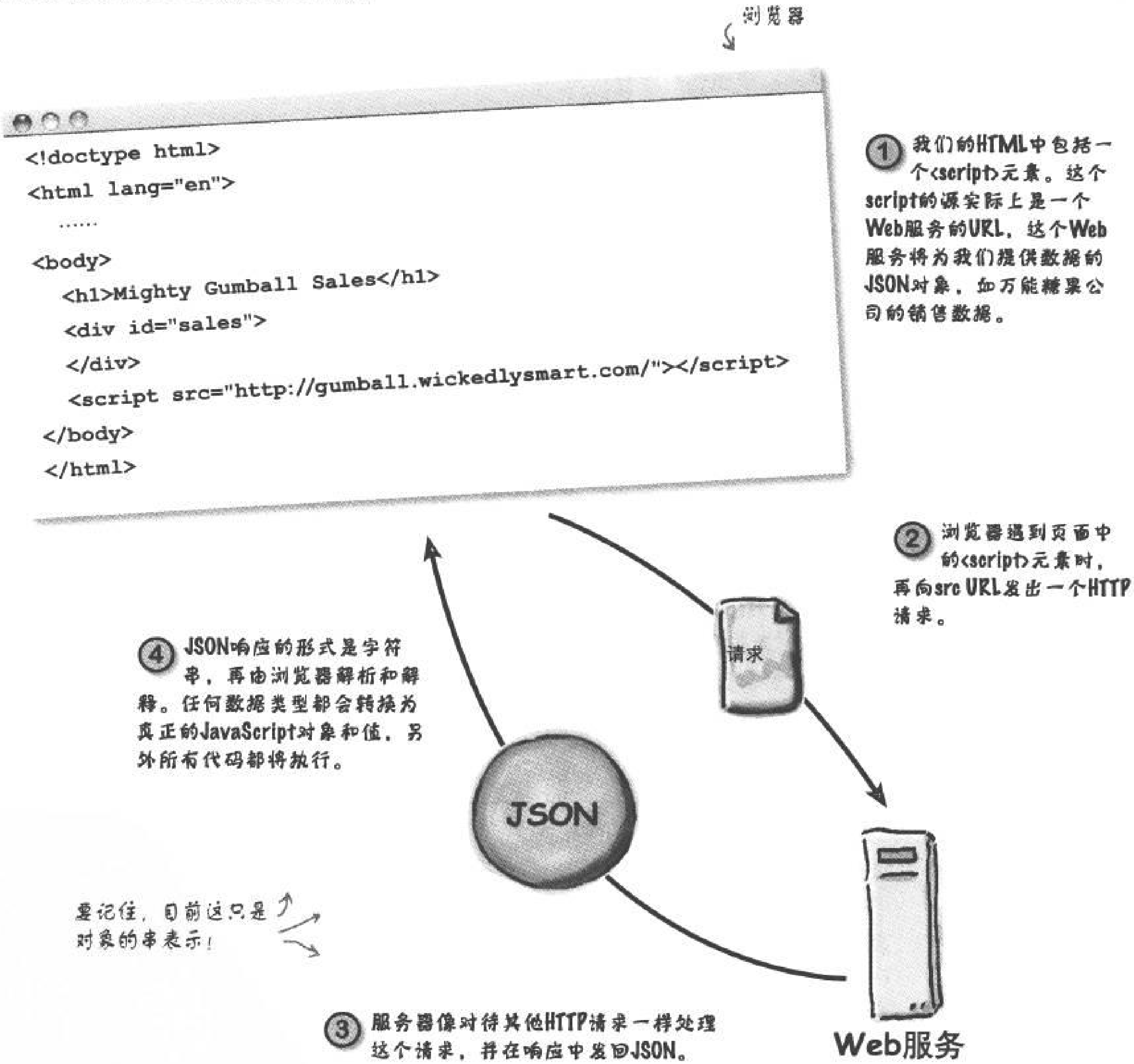
要想有点成就，你必须学会使用JavaScript获取数据。拿出一张纸，或者可以用这本书的书皮背面。画出一个服务器，存放你的HTML & JavaScript文件。另外在另一个域画一个服务器，其中存放文件dog3.js和cat3.js。现在来完成浏览器得到和使用各个文件中的对象时所用的各个步骤。等你觉得自己确实掌握了，我们再一起来完成这个过程。

pin51.com 拼吾爱

认识JSONP

你已经了解了，JSONP是一种使用<script>标记获取JSON对象的方法。这也是一种获取数据的方法（同样的，采用JSON对象的形式），它可以避免XMLHttpRequest的同源安全问题。

下面几页逐步分析JSONP是如何工作的：

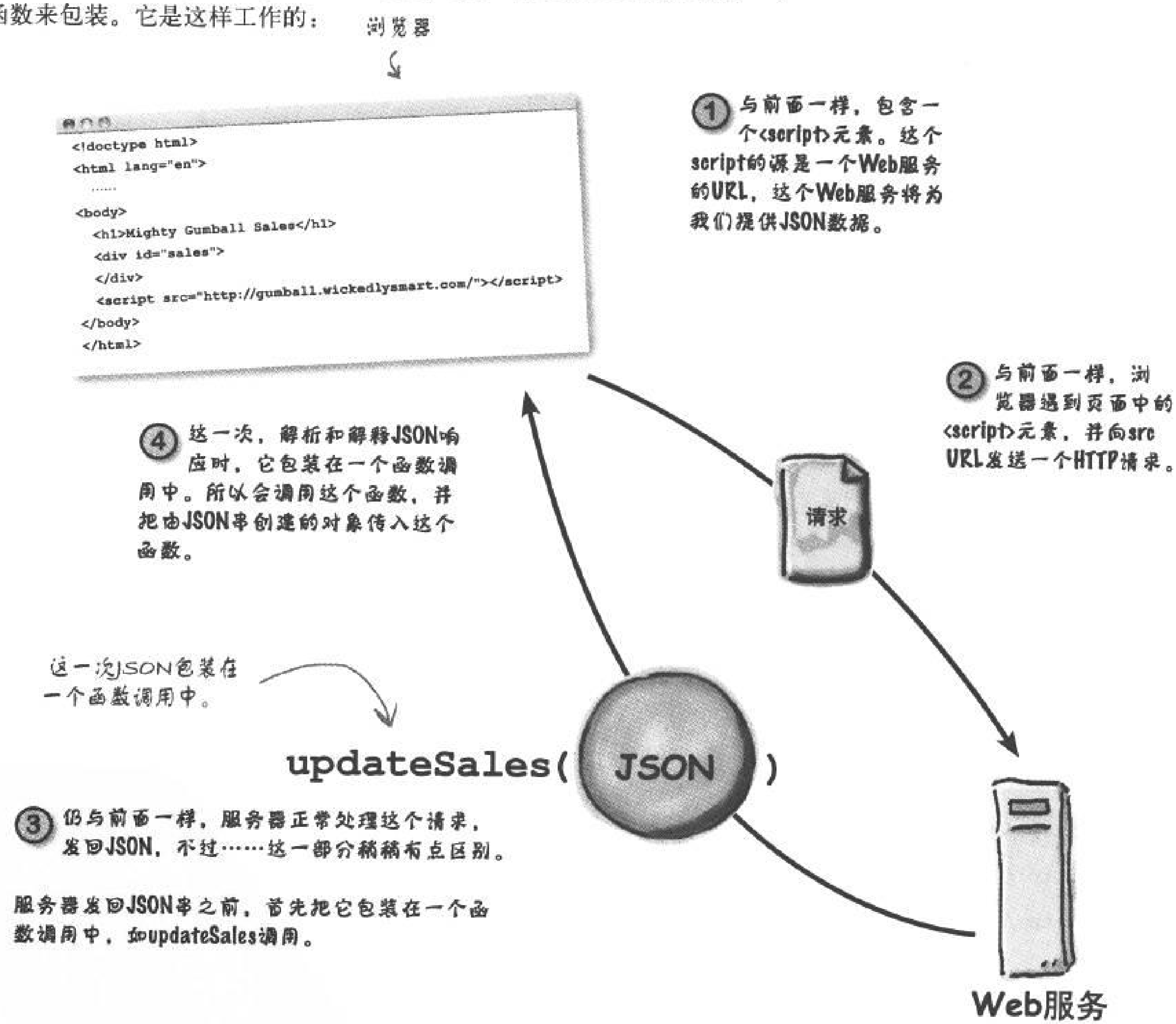


pin51.com 拼吾爱 最新资源下载站

那么JSONP中的“P”代表什么？

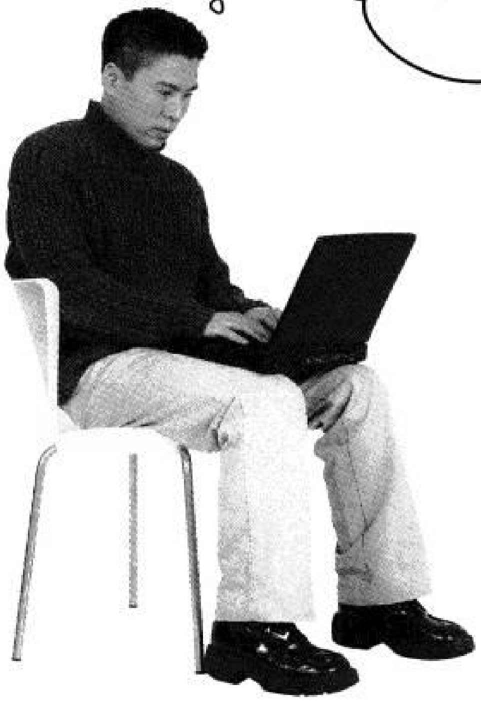
嗯，对于JSONP，首先需要知道，它有一个让人不太明白的名字：“JSON with Padding”（有填充的JSON）。如果让我们来取名，我们会把它叫做“JSON with a Callback”（有回调的JSON）或者“给我一些JSON，等你返回时再执行”，或者，实际上可以是除了“JSON with Padding”以外的任何名字。

不过，这里填充（padding）的含义就是在请求中返回JSON之前先用一个函数来包装。它是这样工作的：



pin51.com 拼吾爱 最新最全资源分享下载站

我要看看如何使用<script>标记让浏览器获取JavaScript，另外服务器怎么把这个数据放在JavaScript中。不过，函数名呢？Web服务怎么知道正确的函数名是什么？比方说，万能糖果Web服务怎么知道要调用updateSales？如果我有另外一个服务，我想让它调用一个函数，比如updateScore或者alert，或者其他什么函数，怎么做到呢？



Web服务允许你指定一个回调函数。

一般来讲，Web服务允许你指定希望如何命名函数。虽然我们没有告诉你，不过万能糖果公司已经支持这样一种方法。它的做法如下：指定URL时，在末尾增加一个参数，就像这样。

```
http://gumball.wickedlysmart.com/?callback=updateSales
```

这是我们一直使用的正常URL。

这里增加了一个URL参数callback，指示生成JavaScript时要使用函数updateSales。

MightyGumball向你发回JSON对象之前，会用`updateSales`包装这个JSON格式化对象。一般地，Web服务将这个参数命名为`callback`，不过请查看你的Web服务文档，明确Web服务实际使用的参数名。

BRAIN POWER

尝试以下URL：响应中会看到什么？

```
http://search.twitter.com/search.json?q=hfhtml5&callback=myCallback
http://search.twitter.com/search.json?q=hfhtml5&callback=justDoIt
http://search.twitter.com/search.json?q=hfhtml5&callback=updateTweets
```

说明：Firefox会要求你打开或保存一个文件。你可以用TextEdit、Notepad或者任何基本文本编辑器打开。

pin51.com 拼吾爱



Jim: 嗯，几乎可以这么说。

Joe: 我认为这实际上允许我们删除一些代码。

Frank: 等你完成之后，我打算让它变漂亮些。

Jim: Joe，你是代码高手，是不是有什么想法了？

Joe: 用XMLHttpRequest时我们会得到一个字符串。使用JSONP时，script标记会解析并执行返回的代码，所以等我们处理数据时，它已经是一个JavaScript对象了。

Jim: 没错，使用XMLHttpRequest时，我们要用JSON.parse把字符串转换为一个对象。现在是不是可以把它去掉了？

Joe: 对。这就是我的想法，我一直这么想。

Jim: 还有别的吗？

Joe: 很显然我们需要插入`<script>`元素。

Jim: 这里我有些不清楚。要把它放在哪里呢？

Joe: 嗯，浏览器在加载页面时会进行控制，我们希望先加载页面，这样调用`updateSales`时就能更新DOM。要处理这个问题，我能想到的唯一办法就是把`<script>`放在页面HTML体的最下面。

Jim: 对呀，听上去是个好想法。我们应该再深入研究研究。不过，作为初学者，先来试试吧。

Joe: OK，我真希望这个代码能正常运行！输入代码吧！

Frank: 你们最好快点，我打赌Judy已经完成她的版本了。

pin51.com 拼吾爱 最新最全资源分享下载站

更新万能糖果公司的Web应用

现在来用JSONP更新你的万能糖果代码。除了删除现有的处理XMLHttpRequest调用的代码外，并没有太大的修改。现在就来完成这些修改：

我们需要做的事情：

1. 删除XMLHttpRequest代码。
2. 确保updateSales函数准备接收一个对象，而不是一个字符串（与XMLHttpRequest不同）。
3. 增加<script>元素，具体完成数据获取。

---

1. onload函数中的所有代码都与XMLHttpRequest有关，所以可以把它完全删除。不过我们还是保留了这个onload函数，以备以后还需要。对于现在来说，这个函数什么也不做。打开mightygumball.js文件，做以下修改：

```
window.onload = function() {
    var url = "http://gumball.wickedlysmart.com";
    var request = new XMLHttpRequest();
    request.open("GET", url);
    request.onload = function() {
        if (request.status == 200) {
            updateSales(request.responseText);
        }
    };
    request.send(null);
}
```

现在直接删除这个函数中的所有代码。

pin51.com 拼吾爱 最新最全电子书

② 接下来，应该记得，使用<script>元素时就是在告诉浏览器需要获取JavaScript，完成解析并执行。这说明，它执行到你的updateSales函数时，JSON已经不再是字符串形式了，而是一个真正的JavaScript对象。我们原先使用XMLHttpRequest时，数据采用字符串的形式返回。现在updateSales会认为它得到的是一个串，所以下面做些修改，让它处理一个对象，而不是字符串：

```
~~function updateSales(responseText) {~~
function updateSales(sales) {
    var salesDiv = document.getElementById("sales");
    ~~var sales = JSON.parse(responseText);~~
    for (var i = 0; i < sales.length; i++) {
        var sale = sales[i];
        var div = document.createElement("div");
        div.setAttribute("class", "saleItem");
        div.innerHTML = sale.name + " sold " + sale.sales + " gumballs";
        salesDiv.appendChild(div);
    }
}
```

删除responseText，重写这行代码，参数名改为sales。

还可以删除JSON.parse调用。

仅此而已，现在已经有了一个可以处理数据的函数了。

③ 最后，增加<script>元素来完成具体的数据获取。

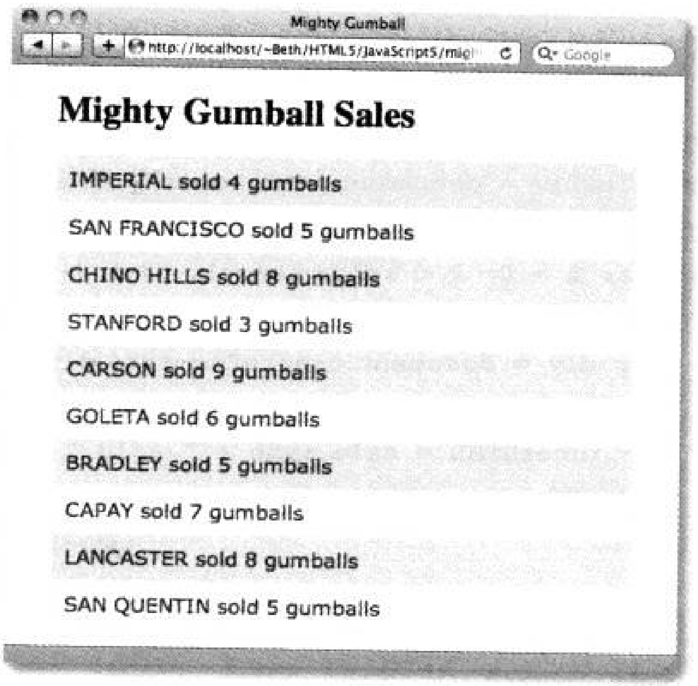
```
<!doctype html>
<html lang="en">
<head>
  <title>Mighty Gumball</title>
  <meta charset="utf-8">
  <script src="mightygumball.js"></script>
  <link rel="stylesheet" href="mightygumball.css">
</head>
<body>
  <h1>Mighty Gumball Sales</h1>
  <div id="sales">
  </div>
  <script src="http://gumball.wickedlysmart.com/?callback=updateSales"></script>
</body>
</html>
```

这是指向万能糖果Web服务的链接。我们使用了callback参数，并指定了我们的函数updateSales，所以这个Web服务会把JSON包装在updateSales函数调用中。

试一试你的JSONP代码



如果你完成了以上的所有修改，现在就来试一试。在浏览器中重新加载mightygumball.html。你现在是用你的Web应用和JSONP加载万能糖果公司的销售数据。页面应该与从本地文件获得销售数据时看到的页面相同，不过你很清楚，目前在用完全不同的方法获取数据。



重新加载万能糖果公司页面会看到这个结果。你看到的可能是不同的城市和销售情况，因为这些是真实数据。





pin51.com 拼吾爱





使用<script>加载JavaScript就不再有安全了。

这话没错：如果向一个恶意Web服务发出一个JSONP请求，响应中可能会包含你不想要的JavaScript代码，而且浏览器会执行这些恶意的代码。

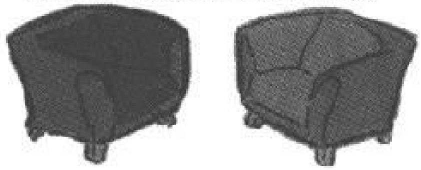
不过这与链接其他服务器上托管的库来包含JavaScript并没有不同。只要链接到JavaScript，不论是在文档的`<head>`中链接到一个库，还是使用JSONP，都需要确保信任这个服务。如果你要编写一个Web应用，其中使用认证来授权用户访问敏感数据，可能最好的办法是根本不要使用第三方库或位于其他服务器的JSON数据。

所以要谨慎选择所链接的Web服务。如果你在使用像Google、Twitter、Facebook或很多其他知名Web服务的API，那么你是安全的。否则，最好当心一点。

对于我们的情况，我们私底下认识万能糖果公司工程师，而且知道他们绝对不会在JSON数据中加入任何恶意的代码，所以我们是安全的，可以继续。

pin51.com 拼吾爱

Fireside Chats



今晚话题：XMLHttpRequest和JSONP

今晚我们请来了从浏览器获取数据的两种流行方法。

XMLHttpRequest: 我没有冒犯你的意思，你不过是耍小聪明吧？我的意思是说，你的目的本来是获取代码，人们却用你来处理数据请求。

JSONP: 小聪明？我倒觉得这应该叫精巧。既然可以用同样的方法来获取代码和数据，何必还要两种单独的方法呢？

XMLHttpRequest: 不过，你所做的无非是在代码里加一些数据而已。你并没有办法直接从JavaScript代码做出请求，还得使用一个HTML <script>元素。这会让你的用户很困惑。

JSONP: 嘿，不管怎么样，这个办法是可行的，人们可以编写代码从服务获取JSON，比如Twitter和Google，以及很多其他的服务。而你呢？你是有安全限制的，所以利用XMLHttpRequest可做不到这些。我是说，你还在活在过去的岁月里，固步自封，还坚守着“XML”，是吧。

XMLHttpRequest: 嘿，XML使用还很广泛，不要诋毁它。用我一样能很好地获取JSON。

JSONP: 当然，如果你希望一直用JSON.parse解析结果。

XMLHttpRequest: 至少，用我就能控制把哪些数据解析为JavaScript。如果用你，可没办法控制，不管怎样都会成为JavaScript。

JSONP: 这应该算是个优点。我的用户得到数据时，都已经为他们解析好了。实际上，我很尊重你，你费了老大劲编写应用来做到这一点，不过问题是你太过苛刻了。在今天的这个Web服务世界里，我们要能向其他的域发出请求。

XMLHttpRequest: 嗯，你可以采用一种技巧来做到，比如JSON-With-Padding。呵，真是个怪名字。或者，你也可以使用正当的方法，这就是XMLHttpRequest，并随它的成长而成长。毕竟，人们一直在努力让我在保证安全的同时更为灵活。

JSONP: 当然了，人们确实是在努力寻求新方法，不过我的用户可等不及了，他们现在就需要。他们可没办法等你慢慢解决所有跨域问题。

pin5i.com 拼吾爱

JSONP: 而且“padding”（填充）一点都不怪，它的意思是，用户做出一个Web服务请求时，还会要求为结果增加一个小小的前缀，比如“updateSales()”。还记得吧，原先人们是怎么称呼你的？Ajax？像不像个浴室清洁器？

XMLHttpRequest: 我和Ajax这个名字没有任何关系，所以不要问我！另外，你是不是从来没有提过你安不安全？

JSONP: 编码人员必须很谨慎。如果你在从另一个服务器获取代码，当然需要知道你在做些什么。不过，你的做法似有不妥，对于这种请求，不能只是回答“不要那么做”。

XMLHttpRequest: 我所说的是，如果你不需要得到其他人的数据，比如Twitter或Google，而且在编写你自己的Web服务和客户，就可以用我。我会更安全，而且使用起来也更直接。

JSONP: 拜托！如今还会有人编写不使用外部数据的服务？听说过“mashup”（混合）吗？

XMLHttpRequest: 那当然，不就是混杂嘛！

JSONP: 嘿，别用这种口气，至少不论在哪里我都能得到一致的支持，我真不喜欢编写那些应付老浏览器的XMLHttpRequest代码。

XMLHttpRequest: 喂，即使老到IE5，要让它支持我并不需要写多少代码。

JSONP: 哈哈，对我来说可是什么代码都不用写的。只需要一个简单的HTML标记。

XMLHttpRequest: 没错，不过不只是这些吧，你试过做重复的事情吗？要知道有时候得反复获取一些东西。比如他们正在处理的万能糖果数据。怎么做到呢？

JSONP: 嘿，没那么糟吧。要处理另一个请求只需要在DOM中写一个新的`<script>`元素。

XMLHttpRequest: 如果我是你的读者，听你这么说我的第一印象就是“你在说什么呢？”

HEAD FIRST: 感谢大家热烈的讨论！很遗憾，时间到了！

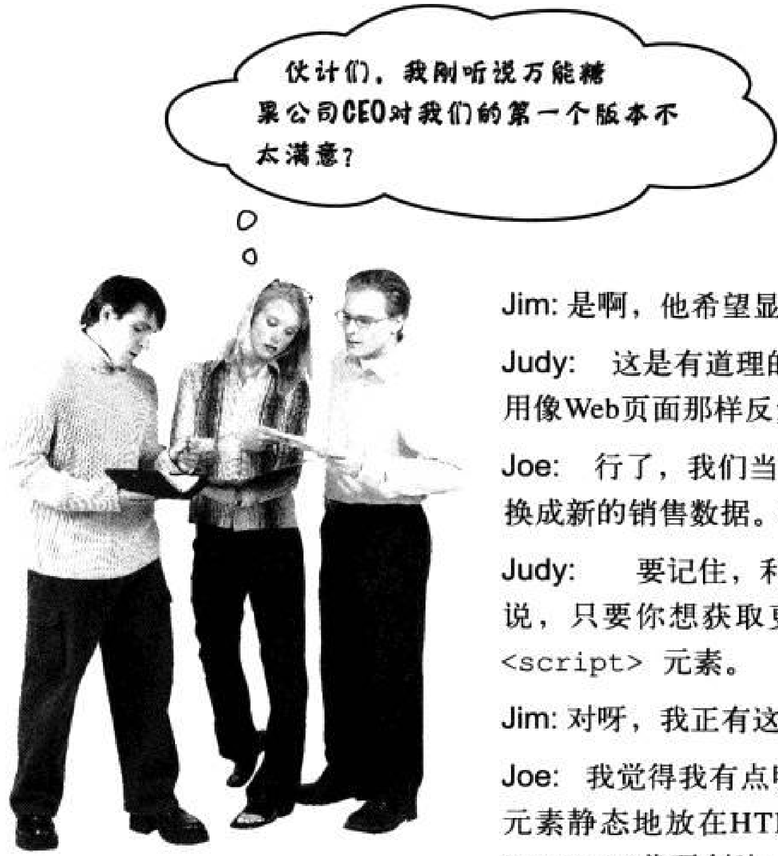


BRAIN POWER

他说的对，我们还要修改我们的应用，让它能定期以某个间隔（比如说，每10秒）根据新的销售数据更新显示。目前我们只是页面中放置了一个<script>元素，它只向服务器发出一次请求。你能想个办法使用JSONP连续地获取新销售报告吗？

提示：使用DOM，我们可以向页面插入一个新的<script>元素。这会有帮助吗？

pin51.com 拼吾爱 最新最全资源分享下载站



Jim: 是啊，他希望显示中数据能不断更新。

Judy: 这是有道理的，我的意思是，Web应用的一大优点就是不用像Web页面那样反复刷新。

Joe: 行了，我们当然知道如何用DOM将页面中的老销售数据替换成新的销售数据。不过还不太确定怎么处理JSONP。

Judy: 要记住，利用`<script>`元素也可以使用DOM。换句话说，只要你想获取更多数据，就可以在DOM中创建一个新的`<script>` 元素。

Jim: 对呀，我正有这个想法。能不能再讲讲？

Joe: 我觉得我有点明白了。目前我们只是通过键入把`<script>`元素静态地放在HTML标记中。可以不这样，我们完全可以用JavaScript代码创建一个新`<script>`元素，并把它增加到DOM。但有一点我不能确定，创建这个新的`<script>`元素时浏览器会完成下一次获取吗？

Judy: 当然会。

Jim: 我明白了，这么说，只要我们希望浏览器为我们完成一个JSONP类型操作，就可以创建一个新的`<script>`元素。

Judy: 太对了！听上去你确实懂了。那么你知道怎么反复做这件事吗？

Jim: 嗯，这个我们还没有搞清楚，我们还在考虑JSONP呢。

Judy: 现在你们已经了解处理函数，比如`onload`或`onclick`。可以建立一个定时器，在JavaScript中使用setInterval方法以一个指定的间隔调用一个处理函数。

Joe: 好的，现在就来建立这个定时器吧，尽快让万能糖果公司CEO看到正常工作的动态JSONP。

Jim: 噢，你也这么想？那么现在就动手吧！

pin51.com 拼吾爱 最新最全电子书下载站

改进万能糖果公司的Web应用

可以看到，我们还有一些工作要做，不过情况不算太糟糕。实际上，我们已经编写了我们的第一个版本，它能从万能糖果公司得到最新的销售报告并显示，不过只是一次。这可不好，因为如今几乎所有Web应用都会持续不断地监视数据，并且（近）实时地更新应用。

我们需要做的事情：

1. 需要从万能糖果HTML删除JSONP <script>元素，因为不再使用这个元素了。

2. 需要建立一个处理程序，每隔几秒就做出一个JSONP请求。我们会采纳Judy的建议，使用JavaScript的setInterval方法。

3. 然后需要在这个处理程序中实现JSONP代码，使得每次调用这个处理程序时会发出请求，来得到最新的万能糖果公司销售报告。

第1步：处理script元素……

我们要用一种新方法调用JSONP请示，所以从HTML中删除原来的<script>元素。

```
<!doctype html>
<html lang="en">
<head>
  <title>Mighty Gumball</title>
  <meta charset="utf-8">
  <script src="mightygumball.js"></script>
  <link rel="stylesheet" href="mightygumball.css">
</head>
<body>
  <h1>Mighty Gumball Sales</h1>
  <div id="sales">
  </div>
  <script src="http://gumball.wickedlysmart.com/?callback=updateSales"></script>
</body>
</html>
```

可以从你的HTML文件中删除这个元素。

pin51.com

第2步：现在来建立定时器



好的，我们从只获取一次销售报告起步，向每隔一段时间就获取一次销售数据前进，比如说每隔3秒。对于不同的应用，这可能太快或者太慢，不过对于万能糖果公司，我们先来考虑间隔3秒时间。

要想每隔3秒做某件事情，需要有一个能够每3秒调用一次的函数。正像Judy提到的，可以使用window对象的setInterval方法来做到；我们可以这样做：

```
setInterval(handleRefresh, 3000);
```

setInterval方法有两个参数：一个处理程序和一个时间间隔。

这是我们的处理函数，稍后来定义。

这是时间间隔，单位为毫秒。3000毫秒 = 3秒。

所以每隔3000毫秒JavaScript就会调用你的处理程序，在这里就是要调用handleRefresh函数。下面写一个简单的处理程序来试一试：

```
function handleRefresh() {
   alert("I'm alive");
}
```

每次调用这个函数时（每隔3秒调用一次），我们会给出一个提醒"I'm alive"（我还活着）。

现在只需要一些代码建立setInterval调用，这些代码要增加到onload函数，整个页面加载之后就会立即建立这个调用：

```
window.onload = function() {
    setInterval(handleRefresh, 3000);
}
```

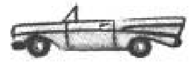
这还是原来的onload函数，删除XMLHttpRequest代码之后它什么也不做。

我们要做的就是增加setInterval调用，运行init函数时，它会启动一个定时器，每3秒触发一次，并调用我们的函数handleRefresh。

下面来试一试，确信它能正常工作。也就是说，等我们看到每3秒调用一次处理程序时，再来实现JSONP代码。

pin51.com 拼吾爱 最新最全资源分享下载站

时间驱动的测试



这应该很有意思。确保你已经键入了handleRefresh函数，而且对onload处理程序做了修改。保存所有修改，并加载到浏览器。你会看到一串提醒，你必须关闭浏览器才能让它停下来！



这就是我们得到的结果！

Sharpen your pencil

既然已经了解了setInterval（当然还有XMLHttpRequest和JSONP），想想看在其他Web应用中还能怎样使用。请写在下面：

检查和更新一个任务的完成进度，并显示。

查看某个主题是否提交了新评论。

如果有朋友在附近出现，就更新地图。



pin51.com 拼吾爱 最新最全的资源分享下载站

第3步：重新实现JSONP

我们仍然希望使用JSONP获取我们的数据，不过需要一种合适的方法，只要调用了刷新处理程序就会获取数据，而不是只在页面加载时才获取。这里就要用到DOM。DOM的过人之处就在于，我们可以在任何时间向DOM插入新元素，甚至是<script>元素。所以，只要想做出一个JSONP调用，就能插入一个新的<script>元素。下面利用我们对DOM和JSONP所掌握的知识，编写一些代码来做到这一点。

首先，建立JSONP URL

这与前面使用script元素时所用的URL完全相同。在这里我们把它赋给一个变量，以备以后使用。删除处理程序中的alert调用，并增加以下代码：

回到我们的handleRefresh函数。

这里，我们建立了JSONP URL，并把它赋给变量url。

```
function handleRefresh() {
    var url = "http://gumball.wickedlysmart.com?callback=updateSales";
}
```

接下来，创建一个新的script元素

现在，不再将<script>元素放在HTML中，要使用JavaScript建立一个<script>元素。需要创建这个元素，然后设置它的src和id属性：

```
function handleRefresh() {
    var url = "http://gumball.wickedlysmart.com?callback=updateSales";

    var newScriptElement = document.createElement("script");
    newScriptElement.setAttribute("src", url);
    newScriptElement.setAttribute("id", "jsonp");
}
```

首先，创建一个新的script元素……

……将这个元素的src属性设置为我们的JSONP URL。

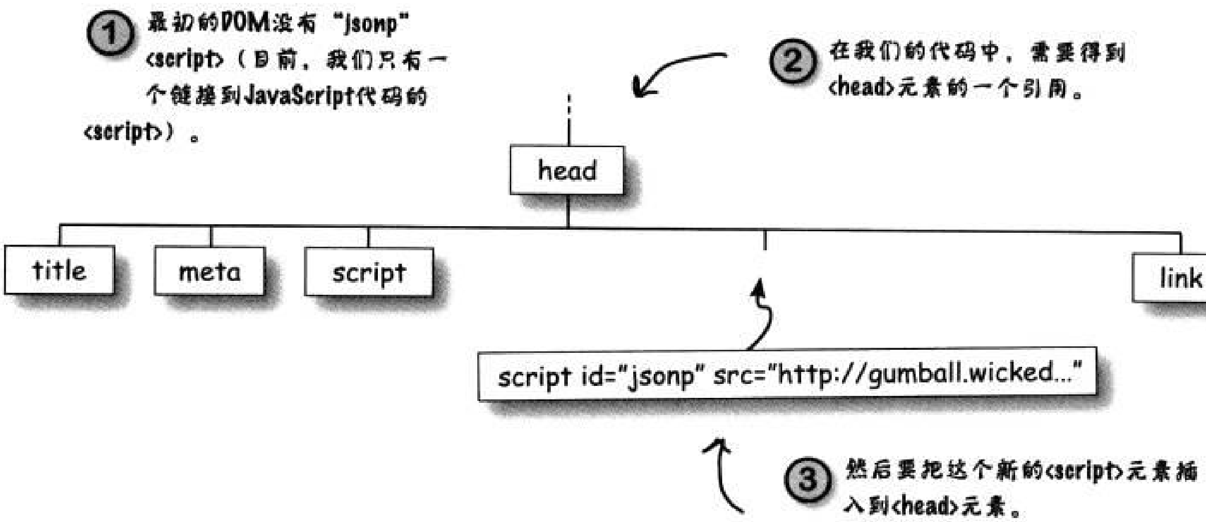
另外为这个script元素指定一个id，从而能容易地得到这个元素。后面会看到，我们确实需要得到这个元素。

对你来说，setAttribute方法看上去可能有些陌生（到目前为止，我们只是简单地提到过），不过很容易看出它要做什么。setAttribute方法允许你设置一个HTML元素的属性，如src和id属性，或者是另外一些属性，包括class、href等。

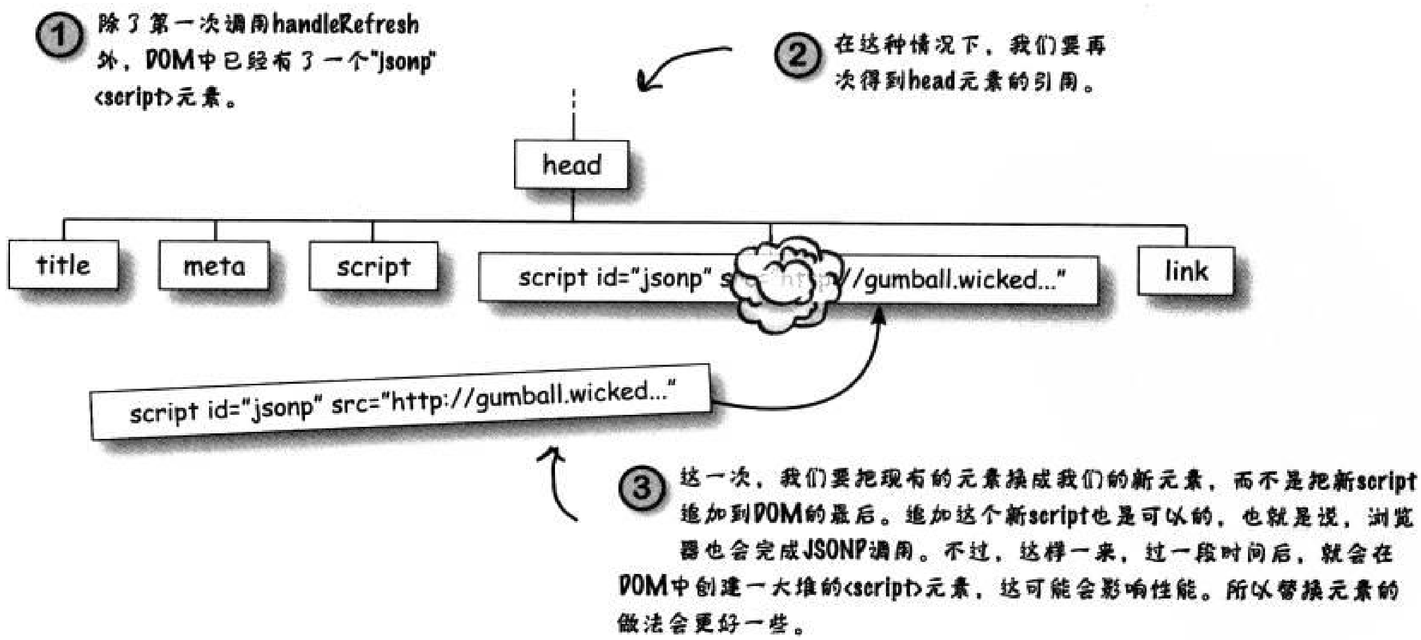
pin5i.com 拼吾爱

如何在DOM中插入script?

就快完成任务了，现在只需要插入新创建的script元素。一旦完成插入，浏览器就会看到它，并完成相应的工作，发出JSONP请求。现在，要插入script还需要预先做些计划，下面来看如何做到：



一旦插入script，浏览器就会看到DOM中的这个新script元素，并获取其src属性指示的URL上的内容。我们还有另一个用例。下面就来看看这个用例。



pin51.com 拼吾爱 最新最全资源分享下载站

现在编写代码将这个script插入到DOM

既然已经知道步骤了，下面来查看代码。同样有两步：首先给出增加一个新script元素的代码，然后给出替换script的代码。

```
function handleRefresh() {
    var url = "http://gumball.wickedlysmart.com?callback=updateSales";

    var newScriptElement = document.createElement("script");
    newScriptElement.setAttribute("src", url);
    newScriptElement.setAttribute("id", "jsonp");

    var oldScriptElement = document.getElementById("jsonp");
    var head = document.getElementsByTagName("head")[0];
    if (oldScriptElement == null) {
        head.appendChild(newScriptElement);
    }
}
```

首先得到<script>元素的引用。如果不存在，则返回null。注意这里要求元素的id为"jsonp"。

接下来，使用一个新的document方法得到<head>元素的引用。后面还会再来讨论这个方法如何工作，不过现在只需要知道它会得到<head>元素的引用就可以了。

既然有了<head>元素的一个引用，现在查看它是否已经有一个<script>元素，如果没有（其引用为null），就把这个新的<script>元素追加到head。

好了，下面来看如果script元素已经存在时替换script元素的代码。这里只给出if条件语句，所有新代码都出现在这里：

这里还是原来的条件，要记住它只是查看DOM中是否已经存在一个<script>元素。

```
if (oldScriptElement == null) {
    head.appendChild(newScriptElement);
} else {
    head.replaceChild(newScriptElement, oldScriptElement);
}
```

如果head中已经有一个<script>元素，就只是替换这个元素。可以在<head>元素上使用replaceChild方法，传入老的和新的script元素来完成替换。稍后还会更详细地讨论这个新方法。

pin51.com 拼吾爱 最新电子书下载分享网站

getElementsByTagName特写

这是我们第一次见到getElementsByTagName方法，所以下面就来仔细研究研究。这个方法与getDocumentById有些类似，只不过它会返回与一个给定标记名匹配的元素数组。

getElementsByTagName返回DOM中有这个标记的所有元素。

```
var arrayOfHeadElements = document.getElementsByTagName("head");
```

在这里，它会返回一个head元素数组。

一旦得到这个数组，可以使用索引0得到数组中的第一项：

```
var head = arrayOfHeadElements[0];
```

返回数组中的第一个head元素（应该只有一个head元素，是吧？）

现在可以把这两行代码结合起来，如下所示：

```
var head = document.getElementsByTagName("head")[0];
```

只需要一步，先得到数组，再索引数组得到第一项。

在我们的代码示例中，总是使用第一个<head>元素，不过你也可以对任何标记使用这个方法，如<p>、<div>等。通常得到的数组中不只有一个元素。

replaceChild特写

再来看replaceChild方法，因为以前你没有见过这个方法。如果想替换一个元素的子元素，可以对这个元素调用replaceChild方法，同时传入新子元素和老子元素的引用。这个方法会用新元素替换老的子元素。

replaceChild方法告诉<head>元素把它的一个子元素oldScriptElement替换为新子元素newScriptElement。

我们的新<script>元素。

页面中已有的<script>。

```
head.replaceChild(newScriptElement, oldScriptElement);
```

pin51.com 拼吾爱 最新最全资源分享

there are no Dumb Questions

问： 为什么要替换整个<script>元素，难道不能直接替换src属性中的数据吗？

答： 如果只是用新URL替换src属性，浏览器不会把它看作是一个新的script元素，所以它不会发出请求来获取JSONP。要强制浏览器做出请求，必须创建这个全新的script元素。这种技术称为"脚本插入"。

问： 替换子元素时，老的子元素会发生什么呢？

答： 会从DOM中删除。此后发生的一切由你决定：如果你把它的一个引用存储在一个变量中，那么还能继续使用这个元素（只要合理，可以采用任何方法）。不过，如果没有变量保存它的引用，JavaScript运行时库可能最后会回收这个元素在浏览器中占用的内存空间。

问： 如果有多个<head>元素呢？用0索引getElementsByTag返回的数组时，你的代码看起来要求只能有一个head，是吗？

答： 根据定义，一个HTML文件只有一个<head>元素。当然了，可能有人会在一个HTML文件中键入两个<head>元素。在这种情况下，你的结果可能会有所不同（如果没有验证HTML，就会遇到这种情况）。不过，浏览器会一如既往尽其所能地保证正确（也就是说，完全可以信赖浏览器）。

问： 启动间隔定时器之后能让它停止吗？

答： 当然可以了。setInterval方法实际上会返回一个id，标识这个定时器。可以把这个id保存在一个变量中，以后只要想停止这个定时器，就可以把它传递到clearInterval方法。关闭浏览器也能停止定时器。

问： 我怎么知道一个Web服务使用的参数呢？另外怎么知道它是否支持JSON和JSONP？

答： 大多数Web服务会发布一个公开的API，其中包括如何访问这个服务，以及利用这个服务可以做哪些事情。如果你在使用一个商业API，可能需要直接从提供商那里得到这个文档。对于很多公开的API，通常可以通过在网上搜索或者由相关组织的网站开发人员得到你需要的信息。你也可以访问诸如programmableweb.com之类的网站，其中提供了一组API的文档，而且还在不断补充。

问： XMLHttpRequest显然比HTML5要老，那么JSON和JSONP呢？它们是HTML5的一部分吗？使用它们时需要有HTML5吗？

答： 我们把JSON和JSONP称作与HTML5是"同时代的"。尽管它们并非由一个HTML5规范定义，但在HTML5应用中大量使用，它们是构建Web应用的一个核心部分。所以，我们说HTML = 标记 + JavaScript API + CSS时，实际上，JSON和JSONP也是其中很重要的一部分（当然也包括采用XMLHttpRequest的HTTP请求）。

问： 人们还在用XML吗？现在是不是一切都是JSON了？

答： 计算机行业里有一条真理，那就是什么都不会彻底消亡，所以我们相信XML还会存在很长一段时间。另外，我们也承认如今JSON活力四射，很多正在创建的新服务都使用JSON。你会经常看到很多Web服务支持多种数据格式，包括XML、JSON和很多其他格式（如RSS）。JSON有一个优点，它直接基于JavaScript，而且JSONP能帮我们绕过跨域问题。

差点忘了：当心可怕的浏览器缓存



pin51.com 拼吾爱

就快完成了，不过还有一个小细节需要注意，这属于那种“如果以前你从来没做过，你怎么知道如何解决”的问题。

大多数浏览器都有一个有趣的特性，如果你反复地获取同一个URL（比如我们的JSONP请求就是这样），浏览器为了提高效率会把它缓存起来，所以你会反复地得到同样的缓存文件（或数据）。这可不是我们想要的。

很幸运，对于这个问题，有一个像Web一样老的简便“疗法”。我们只需要在URL的末尾增加一个随机数，就会诱使浏览器认为它是以前从来没有见过的一个新URL。下面就来修正我们的代码，将上面的URL代码行改为：

你会在handleRefresh函数的最上面看到这行代码。

把上面的URL声明修改为：

```
var url = "http://gumball.wickedlysmart.com/?callback=updateSales" +
        "&random=" + (new Date()).getTime();
```

我们在URL的末尾增加了一个新的“怪怪的”参数。Web服务器会将它忽略，不过足以骗过浏览器了。

我们创建了一个新的Date对象，使用Date对象的getTime方法来得到时间（单位为毫秒），然后把这个时间增加到URL的末尾。

利用这个新代码，生成的URL可能如下：

这一部分应该很熟悉……

这是random参数。

```
http://gumball.wickedlysmart.com?callback=updateSales&random=1309217501707
```

这一部分每次都会改变，从而能避免缓存。

把`handleRefresh`函数中的url变量声明换成以上代码，然后就可以试一试了！


ONE


TWO
THREE

部学习交流使用，未经允许不得用于任何商业用途。感谢
您的配合！

pin5i.com
拼吾爱
最新電腦資源分享下載站

郑重声明

最值得程序员珍藏的200部技术经典系列仅供程序员内

pin5i.com 拼吾爱

pin5i.com
拼吾爱
最新最全资源下载站

拼吾爱 pin5i.com

pin51.com 拼吾爱

再来试一试（时间）

好了，这一次我们已经考虑周全了。应该一切就绪了。一定要确保键入上一次测试之后完成的所有代码，并且重新加载页面。哇，我们终于看到连续的更新了！



等一下……你和我们看到的是一样的吗？这些重复是怎么回事？这可不好。嗯，是不是我们获取得太快了？得到了之前已经获取的报告？

重复！

如何删除重复的销售报告

如果再简单看看228页上的糖果公司规范，你会看到请求URL中可以指定一个最近报告时间参数，如下：

还可以在URL的末尾增加一个lastreporttime参数，这样只会得到这个时间之后的报告。可以像下面这样使用：

只需指定一个时间（单位为毫秒）。

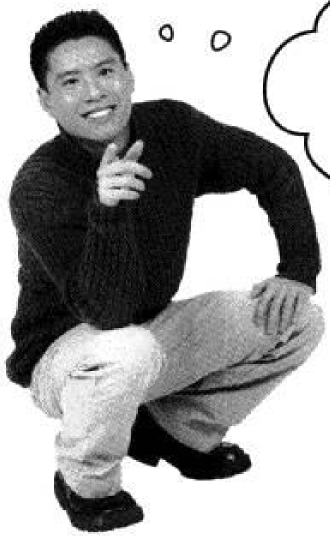
```
http://gumball.wickedlysmart.com/?lastreporttime=1302212903099
```

这很好，但是我们怎么知道获取的最后一个报告的时间呢？再来看看销售报告的格式：

```
[{"name":"LOS ANGELES","time":1309208126092,"sales":2},
 {"name":"PASADENA","time":1309208128219,"sales":8},
 {"name":"DAVIS CREEK","time":1309214414505,"sales":8}
 …]
```

每个销售报告都有报告时间。

pin51.com 拼吾爱



我知道你为什么要用到这个参数：可以跟踪最后一个报告的时间，然后在做出下一个请求时使用这个时间，使得服务器不会返回我们已经获取过的报告，是这样吧？

完全正确。

为了跟踪接收的最后的销售报告，我们需要对`updateSales`函数做一些补充，对销售数据的处理都在这个函数中完成。不过，首先，我们要声明一个变量来保存最后报告时间：

```
var lastReportTime = 0;
```

这个时间不能小于0，所以开始时将它设置为0。

把这个变量声明放在JavaScript文件的最前面，要放在所有函数之外。

下面在updateSales中得到最新销售报告的时间：

```
function updateSales(sales) {
    var salesDiv = document.getElementById("sales");
    for (var i = 0; i < sales.length; i++) {
        var sale = sales[i];
        var div = document.createElement("div");
        div.setAttribute("class", "saleItem");
        div.innerHTML = sale.name + " sold " + sale.sales +
            " gumballs";
        salesDiv.appendChild(div);
    }
    if (sales.length > 0) {
        lastReportTime = sales[sales.length-1].time;
    }
}
```

如果查看sales数组，你会看到最新的销售报告就是数组中的最后一个元素。所以这里将它赋给变量lastReportTime。

不过，要确保确实有一个数组。如果没有新的销售数据，我们会得到一个空数组，这里的代码就会导致一个异常。

pin51.com 拼吾爱

更新JSON URL来包含lastreporttime

既然我们已经跟踪了最新报告的销售数据的时间，要把它作为JSON请求的一部分发送给万能糖果公司。为做到这一点，需要编辑handleRefresh函数，增加lastreporttime查询参数，如下：

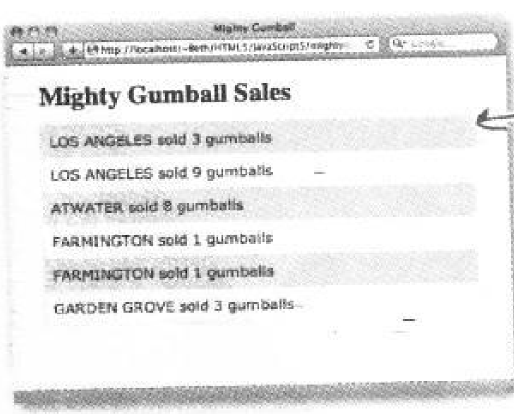
```
function handleRefresh() {
    var url = "http://gumball.wickedlysmart.com" +
              "?callback=updateSales" +
              "&lastreporttime=" + lastReportTime +
              "&random=" + (new Date()).getTime();
    var newScriptElement = document.createElement("script");
    newScriptElement.setAttribute("src", url);
    newScriptElement.setAttribute("id", "jsonp");
    var oldScriptElement = document.getElementById("jsonp");
    var head = document.getElementsByTagName("head")[0];
    if (oldScriptElement == null) {
        head.appendChild(newScriptElement);
    }
    else {
        head.replaceChild(newScriptElement, oldScriptElement);
    }
}
```

我们把URL分解为多个串，以后可以再连接起来……

……这是lastreporttime参数和它的新值。

试一试lastReportTime

下面就来实际试一试lastreporttime查询参数，看看它能不能解决我们的重复销售报告问题。要确保输入新代码，重新加载页面，并点击刷新按钮。



太好了！现在我们只得到了新的销售报告，所有重复的报告都不见了！



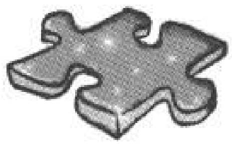
pin51.com 拼吾爱 最新最全资源分享下载站



pin51.com 拼吾爱 最新最全资源分享下载站

BULLET POINTS

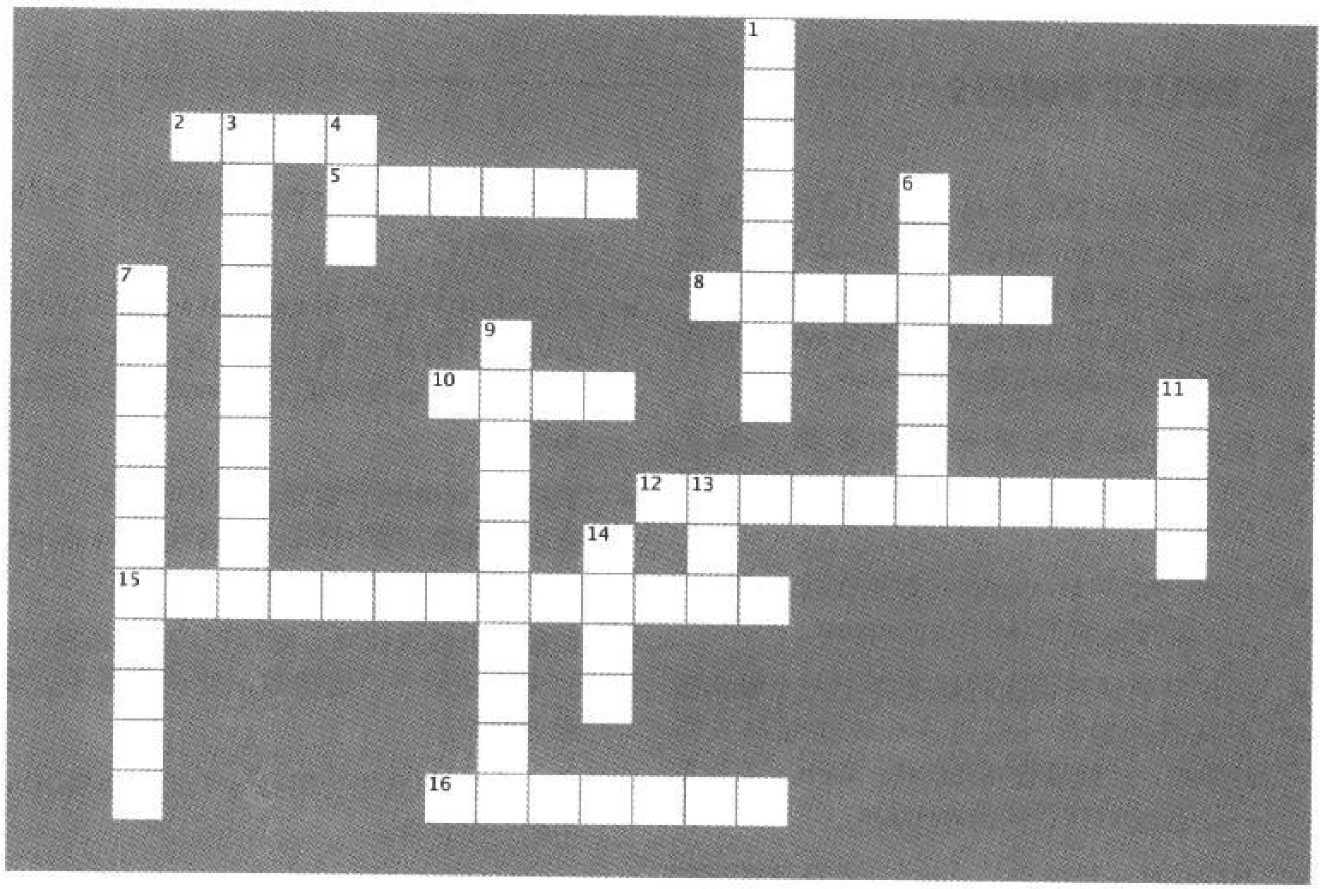
- 除了提供HTML和JavaScript的服务器外，XMLHttpRequest不允许从其他不同服务器请求数据。这是一个浏览器安全策略，专门设计用来避免恶意的JavaScript访问你的Web页面和用户的cookie。
- 要访问Web服务托管的数据，除了XMLHttpRequest外，另一种候选方法是JSONP。
- 如果HTML和JavaScript与数据在同一个机器上，就可以使用XMLHttpRequest。
- 如果需要访问由远程服务器上一个Web服务托管的数据（假设这个Web服务支持JSONP），则要使用JSONP。Web服务就是一个通过HTTP访问的Web API。
- JSONP是一种使用<script>元素获取数据的方法。
- JSONP就是JSON数据包装在JavaScript中，通常会包装在一个函数调用中。
- 将JSON数据包装在JSONP中的函数调用称为“回调”。
- 将回调函数指定为JSONP请求中的一个URL查询参数。
- JSONP并不比使用<script>元素链接到JavaScript库的做法更安全（或更不安全）。只要链接到第三方JavaScript都要特别当心。
- 要指定<script>元素做出JSONP请求，可以把它直接增加到HTML，或者使用JavaScript将<script>元素写至DOM。
- 如果做出多次请求，可以在JSONP请求URL的末尾使用一个随机数，使浏览器不会缓存这个响应。
- replaceChild方法会用另一个元素替换DOM中的一个元素。
- setInterval是一个定时器以指定的间隔调用一个函数。可以使用setInterval向服务器做出重复的JSONP请求来获取新数据。



HTML5填字游戏

pin51.com 拼吾爱 最新最全的资源分享下载站

哇，不得了，这一章已经让你的应用与Web交流了！现在来让左脑活动活动，帮助你把这些内容都牢牢记住。



横向

2. 使用XMLHttpRequest从服务器获取数据的模式有时称为 ______。
5. ______ 是万能糖果公司最新推出支持Web的糖果机。
8. XMLHttpRequest取笑JSONP的__________。
10. 高手教菜鸟函数参数也是 ______。
12. 很 __________，这一章我们讲了25页才发现浏览器安全策略。
15. XMLHttpRequest在Microsoft的一个绰号。
16. JSONP代表“JSON with ______”。

纵向

1. JSONP使用一个 __________。
3. JSONP使用这种类型的对象。
4. 我们原本以为能拯救世界的格式。
6. ______ 有一个JSONP Web服务。
7. 这一章中有这样一个东西。
9. 万能糖果公司在__________测试MG2200。
11. ______提醒Frank、Jim和Joe：XMLHttpRequest存在跨域安全问题。
13. 很容易在 ______上建立一个本地服务器。
14. ______，质量保证检验员，看到向Gumball生产服务器提交的请求失败了很郁闷。

pin51.com 拼吾爱

来自第7章的特殊消息……



虽然我们在让你考虑JSONP，不过我们还希望能在关于画布的那一章得到你的帮助。

我们在利用Twitter的JSONP API构建一个服务，使你能把任何微博印在T恤上。

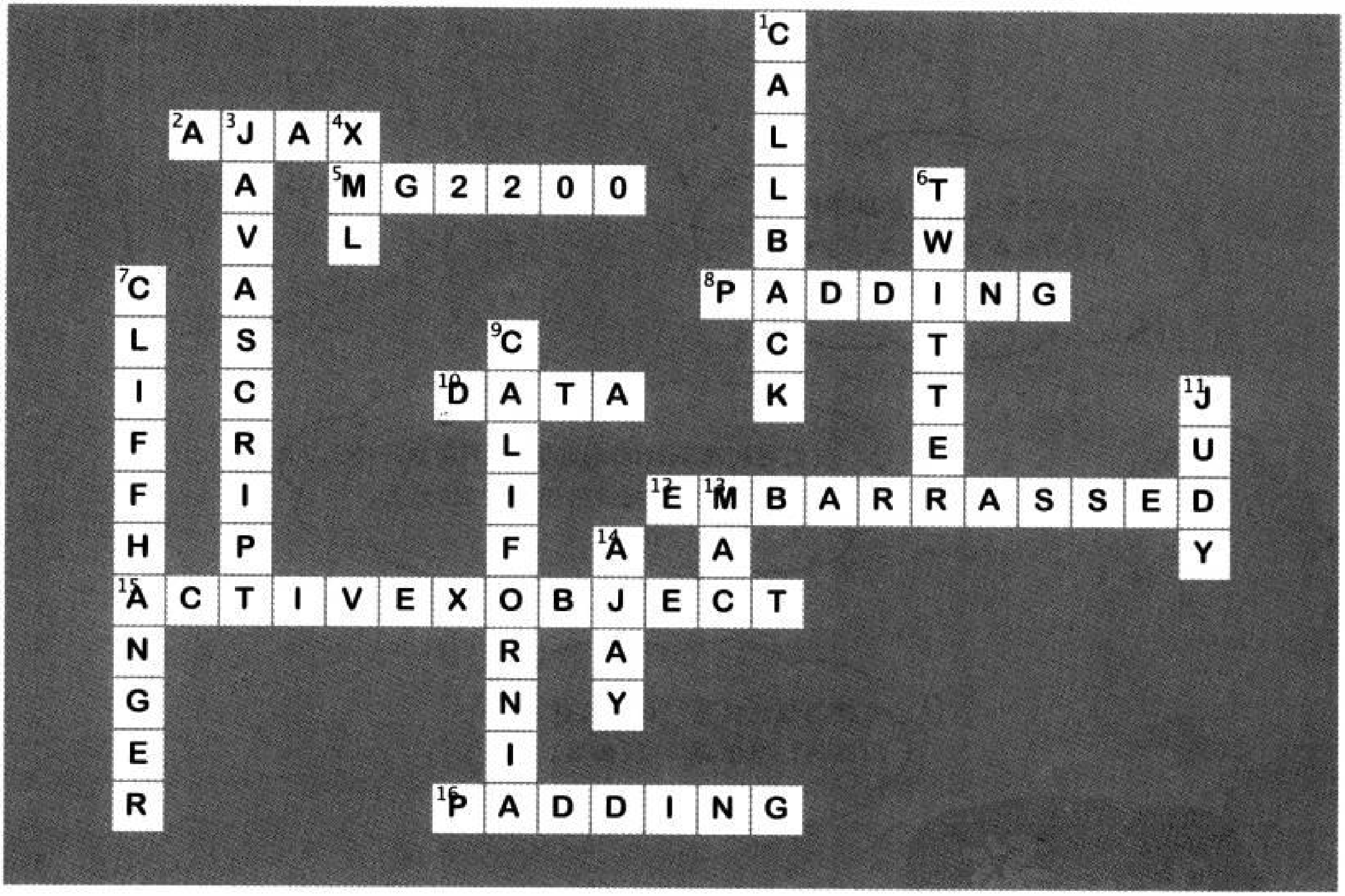
我们想说的是“如果值得发微博，就值得印在T恤上。”



TweetShirt.com创始人。

pin51.com 拼吾爱

HTML5填字游戏答案



7 秀出你的艺术天份

pin51.com 拼吾爱 最新最全资源下载站

画布



HTML已经不再只是一个“标记”语言了。 利用HTML5新增的画布元素，你已经具备亲手创建、管理和消灭像素的能力。这一章中，我们将使用画布元素让你秀出你的艺术天份。别再说什么HTML只有语义而没有表现力，有了画布，我们完全可以绘制出一个多彩的世界。现在就来谈谈表现。我们将处理如何在页面中放置画布，如何绘制文本和图片（当然是使用JavaScript），甚至还会讨论如何处理不支持画布元素的浏览器。另外，画布不会是昙花一现，在这本书的其他章中画布还将屡屡露面。

嗯，“消灭”这个词可能有点杀伤力过大。

实际上我们听说<canvas>和<video>除了共享Web页面之外还有很多共同之处……后面还会讨论这个有趣的内容。

pin51.com 拼吾爱

新的创业项目：TweetShirt



我们的口号是“如果值得在Twitter上发微博，就值得印在T恤上”。

毕竟，如果希望成为一个撰稿人，愿意把你的文字印出来，你就成功了一半，那么如果不印在你自己或别人的胸口上，还有什么更好的地方？无论如何，这是我们的创业产品，就这么办吧。

目前，这个创业项目要上马只有一个障碍：我们迫切需要一个漂亮的Web应用，能够让顾客创建定制的T恤设计，显示他们最新发布的微博。

你可能在想“真的吗？这个主意不坏”。那么，跟我来吧，这一章最后我们就会让这个创业项目上马。噢，对了，如果你真的照这样做了，而且挣了钱，我们并没有申请知识产权，也不要求你交任何费用，不过至少给我们一件免费的T恤呗！



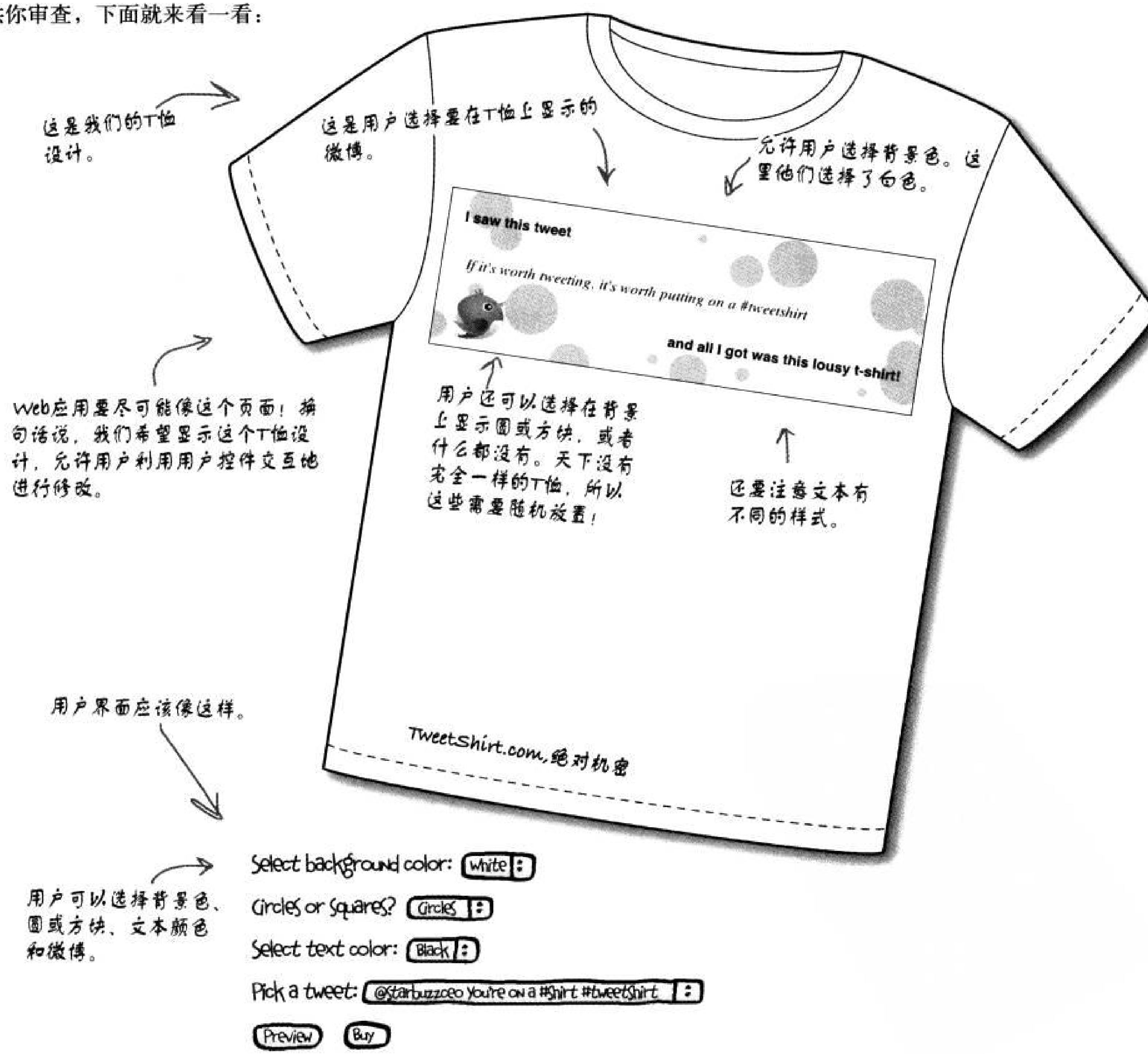
TweetShirt.com创始人。

pin51.com 拼吾爱

审查“初样”

经过详尽的迭代设计，另外在销售组的大量测试之后，我们得到了一个初样（也称为前期形象设计）供你审查，下面就来看一看：

对，来看看吧，我们刚刚创业，这是在星巴克餐巾纸上做的设计。



pin51.com 拼吾爱 最新最全资源下载站

BRAIN POWER

再来看看上一页的需求。你认为可以如何使用HTML5完成这个任务？还记得吗？其中一个需求是：确保你的网站尽可能适用更多的设备格式和大小。

选中以下所有可能的做法（然后圈出最佳答案）：

- ☐ 只使用Flash，它在大多数浏览器上都适用。
- ☐ 可以试试HTML5，看看有没有新技术可以提供帮助（提示：可能有一个称为画布的技术）。
- ☐ 为每一种设备编写一个定制应用，这样一来就能知道具体会得到怎样的用户体验。
- ☐ 在服务器端创建图像，再将一个定制的图像传回浏览器。

there are no Dumb Questions

问：说正经的，对于这个问题，为什么不用Flash，或者一个定制应用呢？

答： Flash是一个很优秀的技术，你当然可以使用这个技术。不过，要知道现在整个行业都在转向HTML5。我们写这本书期间，你要想在所有设备上运行你的Flash应用，可能会遇到麻烦，甚至包括一些非常流行的设备。

如果你真的需要一种完全为设备定制的体验，实现定制应用可能是一个很好的选择。但是要记住，为各种设备分别开发一个定制应用成本可能很昂贵。

利用HTML5，你可以得到移动设备和桌面浏览器的广泛支持，而且通常使用一种技术方案就可以创建普遍适用的应用。

问：我很喜欢在服务器端创建图像的想法。这样一来，我只用写一段代码，图像可以适用所有设备。我懂一点PHP，所以应该能搞定。

答： 这也是一种可选的方法，不过这种方法有一些缺点，如果你有很多很多用户，就必须考虑扩展服务器来满足膨胀的用户需求（与此同时，每个用户的客户端要负责生成T恤的预览视图）。但是，如果为浏览器编写代码，你还能得到更有交互性、更无缝的体验。

怎么做到？嗯，很高兴你这么问……

www.m51.com 最新资源 精品教程

拜访TweetShirt团队……

你已经了解了需求，另外还为用户体验做了一个基本设计，现在来解决最棘手的部分，如何让应用运转起来。下面来听听目前的进展……



Frank, Judy和Joe.

Joe: 看到背景上的圆之前，我还以为这很简单呢。

Frank: 你是什么意思？这只不过是图像而已……

Judy: 不，绝不那么简单。创始人希望这些圆是随机的,所以我的T恤上的圆应该和你的不一样。同样的，不同T恤上的方块也应该不同。

Frank: 没问题，以前我们做到过，只需要在服务器端生成图像就可以了。

Joe: 嗯，我知道，不过这种方法好像不太好；记得吧？这样得向Amazon付大笔的服务器费用。

Frank: 噢，对，不过没关系。

Joe: 不论怎样，我们希望问题能立即解决，要知道，没有那么长时间返回到服务器了。所以如果可以的话，就在客户端完成吧。

Judy: 伙计们，我认为可以做到。我一直在研究HTML5中的画布。

Frank: 画布？要记住，我只是一个设计人员，快跟我说说。

Judy: 你肯定听说过画布，Frank,这是HTML5中的一个新元素，它能创建一个可绘制的区域来放置2D形状、文本和位图图像。

Frank: 听起来就像一个`<img>`标记。只需要放在页面上，指定一个宽度和高度，浏览器就会完成余下的工作。

Judy: 这样比较也不错，我们确实要为画布定义一个宽度和高度，不过在这里，画布上绘制的内容要用JavaScript代码指定。

Joe: 那么，标记怎么处理呢？能在JavaScript中告诉画布，“把这个`<h1>`元素放在这里”？

Judy: 不行，在页面中放入画布之后，你就不要再考虑标记了.在JavaScript中我们会放置点、线、路径、图像和文本。这是一个底层API。

Joe: 哦，只要能画出那些随机的圆，我就很高兴。好吧，说得够多了，下面来具体看看吧！

如何在Web页面中增加画布

Frank说得对，从很多方面讲，画布就像是一个`<img>`元素。可以这样增加一个画布：

canvas（画布）元素是一个正常的HTML元素，首先是一个开始<canvas>标记。

width属性定义它在Web页面中水平方向上占多少个像素。

类似的，height定义了它所占的页面垂直区域，在这里是200像素。

```
<canvas id="lookwhatIdrew" width="600" height="200"></canvas>
```

我们增加了一个id来标识这个画布，稍后会看到如何使用这个id……

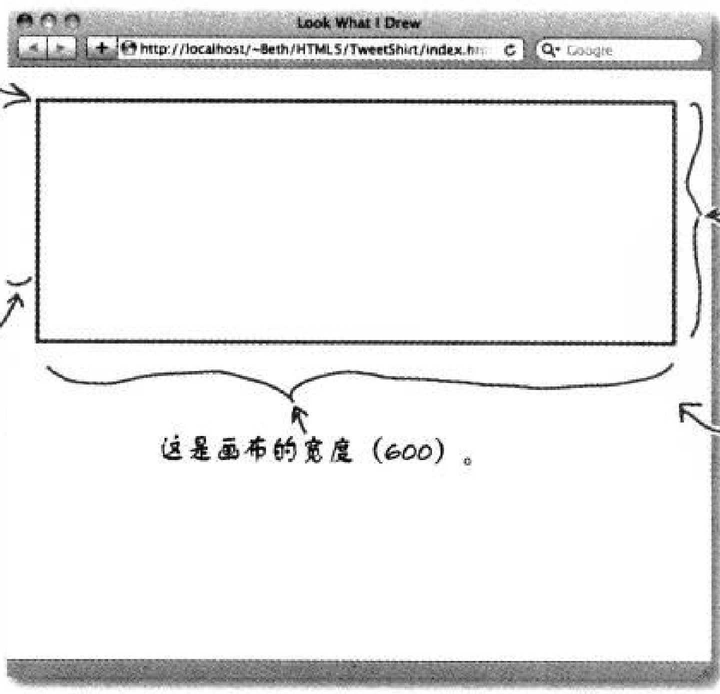
这里width设置为600像素宽。

这里是结束标记。

浏览器会根据指定的宽度和高度在页面中为画布分配一些空间。

在这里，宽度为600，高度为200。

这是画布的左上角。我们使用这个点来度量画布上所有元素的位置（稍后就会看到）。



这是画布的高度（这里为200）。

画布四周有一些空白，这是体元素默认的外边距。

这是画布的宽度（600）。

画布四周可以环绕HTML。画布与其他元素并没有不同（比如图像等）。

pin51.com 拼吾爱 最新资源分享下载站

试一试你的新画布

现在在你自己的Web页面中试一试。把下面的代码键入到一个新文件，然后在你的浏览器中加载这个文件：

```
<!doctype html>
<html lang="en">
<head>
    <title>Look What I Drew</title>
    <meta charset="utf-8">
</head>
<body>

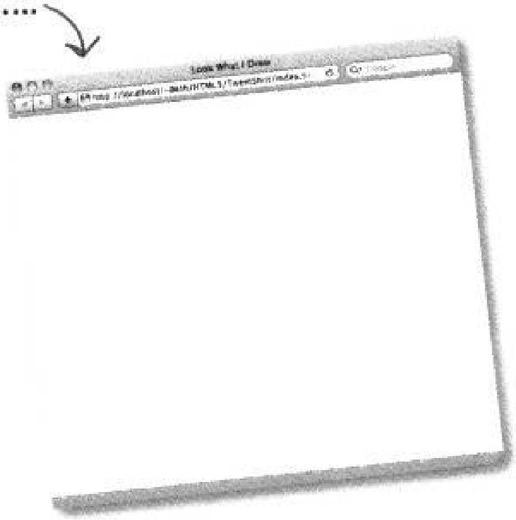
<canvas id="lookwhatIdrew" width="600" height="200"></canvas>

</body>
</html>
```

输入这个代码试一试。



她看到的是……



……你可能也会看到同样的结果！

我们画了这些线来解释画布在页面上如何摆放，这只是为了便于说明。实际上并没有这些线（除非你自己画出）。

翻到下一页了解更多……

pin51.com 拼吾爱

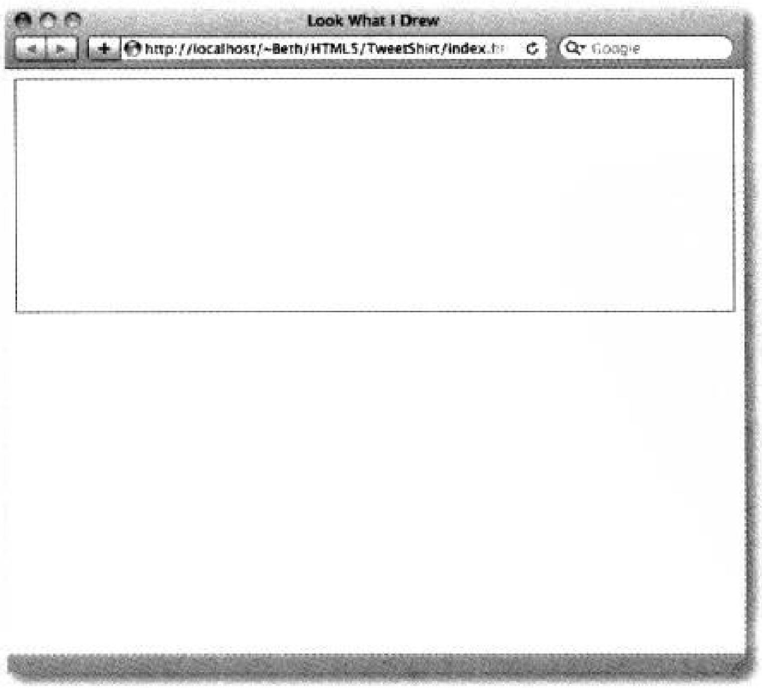
如何看到画布

除非你在画布上绘制了内容，否则你是看不到它的。画布只是浏览器窗口中的一个空间，你可以在其中完成绘制。稍后我们就会在画布中绘制具体内容了，不过现在我们只想找到一个证据，证明画布在我们的页面中确实存在。

还有一种办法可以看到画布…… 如果使用CSS为<canvas>元素指定样式，就能看到边框，这样我们就能在页面上看到画布了。下面就来增加一个简单的样式，为画布增加一个1像素宽的黑色边框。

```
<!doctype html>
<html lang="en">
<head>
    <title>Look What I Drew</title>
    <meta charset="utf-8">
    <style>
      canvas {
        border: 1px solid black;
      }
    </style>
</head>
<body>
<canvas id="lookwhatIdrew" width="600" height="200"></canvas>
</body>
</html>
```

我们为画布增加了一个样式，它在画布上增加了一个1像素宽的黑色边框，这样我们就能在页面上看到画布了。



好多了！现在我们可以看到画布了。接下来，需要对它做些有意思的处理……

www.aibbt.com 爱悠悠·爱好者

there are no Dumb Questions

问：是不是一个页面只能有一个画布？

答：不，你希望有多少个都可以（或者只要不超出浏览器或者你的用户的处理能力）。但是要为每个画布指定一个唯一的id，这样就能将它们作为单独的画布分别完成绘制。稍后你就会看到如何使用画布id。

问：画布是透明的吗？

答：默认情况下是这样的，画布是透明的。可以在画布中进行绘制，填入有颜色的像素，这一章后面会介绍如何做到。

问：如果画布是透明的，这是不是说，我可以把它放在另一个元素的上面，比如在一个图像或页面上另外某个元素的上面进行绘制，是这样吗？

答：非常正确！这正是画布的一个诱人之处。利用画布，你能够在页面上的任何位置增加图片。

问：我可以使用CSS设置画布的宽度和高度吗？而不是利用元素的width和height属性？

答：这是可以的，不过它的做法可能与你想象中稍有出入。默认的，画布元素是300像素宽、150像素高。如果未在canvas标记中指定width和height属性，就会得到这个大小的画布。如果接下来在CSS中指定了一个大小，比如600px × 200px，那么原来的300 × 150的画布就会扩展到这个大小，所以画布中绘制的所有内容也会随之扩展。这就像为一个图像指定新的宽度和高度时，如果比原图像的实际宽度和高度更大或更小时，就会将图像拉伸或缩小。如果将它放大，就会在图像中得到像素化的效果，对不对？

画布也是一样。300px宽的画布变成600px宽时，同样数目的像素会扩展为原来大小的两倍，所以看起来会变得矮矮胖胖。不过，如果使用元素的width和height属性，会把画布的大小设置得比300 × 150更大（或更小），但画布上的所有内容仍会正常绘制。所以我们建议还是在标记属性中指定宽度和高度，而不要在CSS中设置这些属性，除非你本来就打算缩放画布。

BRAIN POWER

你可能已经注意到，画布元素中没有任何内容。如果在开始和结束标记之间放置文本，你认为页面加载时浏览器会怎么做？

<canvas> ? </canvas>

pin51.com 拼吾爱

在画布上绘图



刚才我们得到了一个空画布，这让我们不免大吃一惊。与其呆坐着不知所措，对编写JavaScript毫无想法，我们不如在画布上画上一个漂亮的黑色填充矩形。要画这个矩形，首先要确定画在哪里，另外矩形有多大。能不能放在x=10像素、y=10像素的位置上，另外让它的高度和宽度都等于100像素？来试试看吧。

现在来看完成这个工作的代码：

```
<!doctype html>
<html lang="en">
<head>
    <title>Look What I Drew</title>
    <meta charset="utf-8" />
    <style>
        canvas { border: 1px solid black; }
    </style>
    <script>
        window.onload =function() {
            var canvas = document.getElementById("tshirtCanvas");

            var context = canvas.getContext("2d");

            context.fillRect(10, 10, 100, 100);

        };
    </script>
</head>
<body>
    <canvas width="600" height="200" id="tshirtCanvas"></canvas>
</body>
</html>
```

首先是我们的标准HTML5。

目前保留CSS边框。

这是我们的onload处理程序，页面完全加载之后开始绘制。

要在画布上进行，我们需要画布的一个引用。这里使用getElementById从DOM得到它的引用。

嗯，这有点意思，显然我们需要画布的一个"2d"上下文来具体进行绘制……。

我们使用2d上下文在画布上绘制一个填充的矩形。

这些数字是矩形在画布上的x,y位置。

另外还要有宽度和高度（单位为像素）。

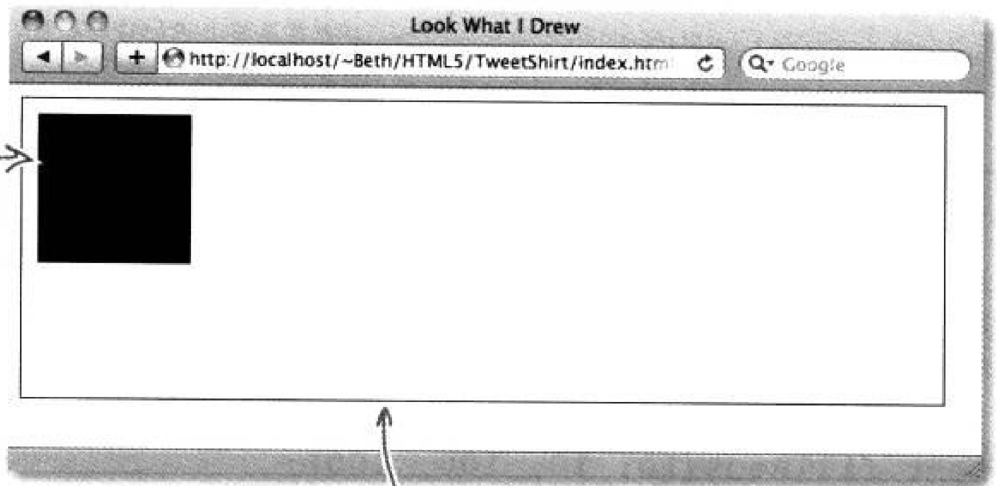
还有一点很有意思，矩形的填充方法居然没有填充色……这个内容稍后再做详细说明。

哈，不能忘了我们的画布元素。我们指定了一个600像素宽、200像素高的画布，id为"tshirtCanvas"。

一个小小的画布测试……

输入这个代码（或者从http://wickedlysmart.com/hfhtml5找到这个代码），把它加载到你的浏览器。假设你在使用一个现代浏览器，应该能看到类似下面的结果：

这是我们的100 x 100矩形，位于画布的10,10位置。



这是我们的画布，600像素宽、200像素高，周围有一个1像素宽的黑色边框。

详细分析代码

这是一个不错的小测试，不过下面我们将更深入地分析这个代码：

1. 我们的标记中使用<canvas>标记定义了一个画布，并指定了一个id。要在这个画布上进行绘制，首先需要从DOM得到这个画布对象的一个句柄。与以往一样，我们用getElementById方法来得到：

```
var canvas = document.getElementById("tshirtCanvas");
```

pin51.com 拼吾爱 最新 下载站 精品资源

2 我们已经将画布元素的引用赋给了`canvas`变量，在画布上具体绘制之前，现在必须达成一个"协议"。我们要请求画布提供一个可供绘制的上下文（context）。在这里，我们明确指出需要一个2D上下文。将画布返回的上下文赋给`context`变量：

```
var context = canvas.getContext("2d");
```

开始在画布上进行绘制之前，这是我们必须遵循的协议。

3 现在，有了上下文，可以用它在画布上进行绘制了，具体的绘制通过调用`fillRect`方法来完成。这个方法会创建一个矩形，以x，y位置(10,10)为起点，宽度和高度都为100像素。

注意，我们在上下文上调用fillRect方法，而不是在画布本身调用这个方法。

```
context.fillRect(10, 10, 100, 100);
```

试试这个代码，你会看到出现一个黑色矩形。试着改变x、y、width和height的值，看看会发生什么。

BRAIN POWER

如果你的浏览器支持画布，你能想办法使用画布元素吗？如果浏览器不支持，能不能显示一个消息？比如"Hey you, yes YOU, upgrade your browser!!"（嘿，你，对，就是你，快点升级浏览器!!）。

拼吾爱

there are no Dumb Questions

问： 画布怎么知道这个矩形是黑色的？

答： 黑色是画布的默认填充颜色。当然，你可以使用fillStyle属性改变默认填充色，稍后我们会介绍。

问： 如果我只想要一个矩形轮廓，不想要一个填充的矩形，该怎么做呢？

答： 如果只想得到一个矩形的轮廓，就要使用strokeRect函数而不是fillRect。本章后面你还会看到更多有关笔划（stroke）的内容。

问： 什么是2d上下文，为什么不能直接在画布上绘制？

答： 画布是Web页面中显示的图形区。上下文是与画布关联的一个对象。它定义了一组属性和方法，可以用来在画布上进行绘制。甚至可以保存上下文的状态，以后再恢复，有时这会很方便。这一章后面你还会看到很多上下文属性和方法。

画布设计用来支持多个接口，除了2d，还有3d，以及我们还没有想到的其他接口。通过使用上下文，就能在同一个画布元素中处理不同的接口。不能直接在画布上绘制，因为你需要选择一个上下文来指定使用哪个接口。

问： 这是不是意味着还有一个“3d”上下文？

答： 还没有。这方面有很多在竞争办的新兴标准，不过目前还没有哪个标准胜出。暂且考虑2d上下文，同时可以看看WebGL和使用这个接口的库，比如SpiderGL、SceneJS和three.js。

正式编码

是不是想知道如何在代码中检测你的浏览器是否支持画布？

你当然能做到，需要指出，到目前为止，我们一直假设浏览器是支持画布的。不过，在生产代码中，确实应该进行测试，以确保浏览器支持画布。

你要做的就是查看canvas对象（这个对象由getElementById返回）中是否存在getContext方法：

首先得到页面中一个画布元素的引用。

```
var canvas =
    document.getElementById("tshirtCanvas");
if (canvas.getContext) {
    // you have canvas
} else {
    // sorry no canvas API support
}
```

然后检查getContext方法是否存在。注意，我们并没有调用这个方法，只是查看它是否有值。

如果你想检查是否提供画布支持，而无需标记中已有一个画布，可以使用你已经了解的所有技术动态地创建一个画布元素。如下所示：

```
var canvas =
    document.createElement("canvas");
```

可以参考附录，来了解一个开源库的有关信息，你可以使用这个开源库采用一种一致的方式检查HTML5中的所有功能。

pin51.com 拼吾爱 最新精品资源分享下载站

我在Internet Explorer中尝试时，在原本该出现画布元素的地方什么都没有看到。怎么回事？



只有IE 9及以后版本才支持画布，所以要适当地编写页面代码，让你的用户了解这一点。

你看这样好不好：如果你实在需要在Internet Explorer（再说一遍，IE 9以前的版本）中支持画布功能，那么请查看Explorer Canvas Project和其他类似的项目，通过这些方法可以使用一个插件来实现这个功能。

不过，对于现在，我们假设你更愿意让你的用户知道他们遗漏了你的绝妙的画布内容。下面就来看看如何做到……

可能你会建议他们升级到IE9！

pin51.com 拼吾爱 最新最全 资源分享下载站

妥善地失败

所以,现在的事实是，总有某个地方、在某个时间，用户可能会访问你的网站，但是没有提供画布元素支持。你是不是想向他们发送一个友好的消息，告诉他们应该升级？可以这样做：

这就是平常的画布元素。

```
<canvas id="awesomecontent">
  Hey you, yes YOU, upgrade your browser!!
</canvas>
```

如果用户的浏览器不支持画布，可以在这里放入希望显示给他们的消息。

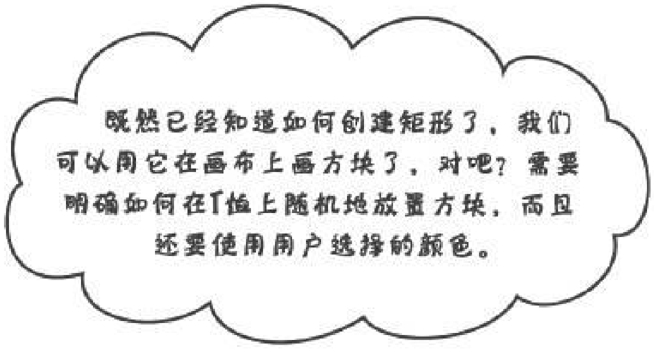
这是怎么做的？嗯，只要浏览器看到一个它不认识的元素，默认行为就是显示其中包含的文本。所以，不支持画布的浏览器看到`<canvas>`元素时，它们就会显示“Hey you, yes YOU, upgrade your browser!!”。而另一方面，支持画布的浏览器会直接忽略画布标记之间的所有文本，所以不会显示这个文本。

谢谢你们，开发HTML5标准的人们，感谢你们让这一切如此容易！



另外，你已经知道，对于不支持画布的浏览器，还有一种处理方法是使用JavaScript来检测浏览器是否认识这个元素。这样可以提供更大的灵活性，能够在用户的浏览器不支持画布时为他们提供不同的体验。例如，可以将他们重定向到一个不同的页面或者转而显示一个图像。

pin51.com 拼吾爱 最新资源分享下载站





Frank: 当然，另外我们还需要一个用户界面，让用户来指定这些选择。我的意思是，虽然我们已经有了“模型”,不过还需要具体实现。

Judy: 你说得对，Frank。如果没有界面，再继续深入也没有太大的意义。

Joe: 那不就是HTML吗？

Frank: 是的，我想是这样。不过，假设这些都在客户端完成，又该怎么做呢？例如，表单在哪里提交？我实在不清楚这些如何集成在一起。

Joe: Frank，我们可以在用户点击预览按钮时调用一个JavaScript函数，然后就能在画布中显示T恤的设计了。

Frank: 有道理，不过如果所有值都在客户端，我们怎么在表单中访问这些值呢？

Judy: 就像访问DOM的方法一样；可以使用`document.getElementById`来获取表单值。你以前这样做过。

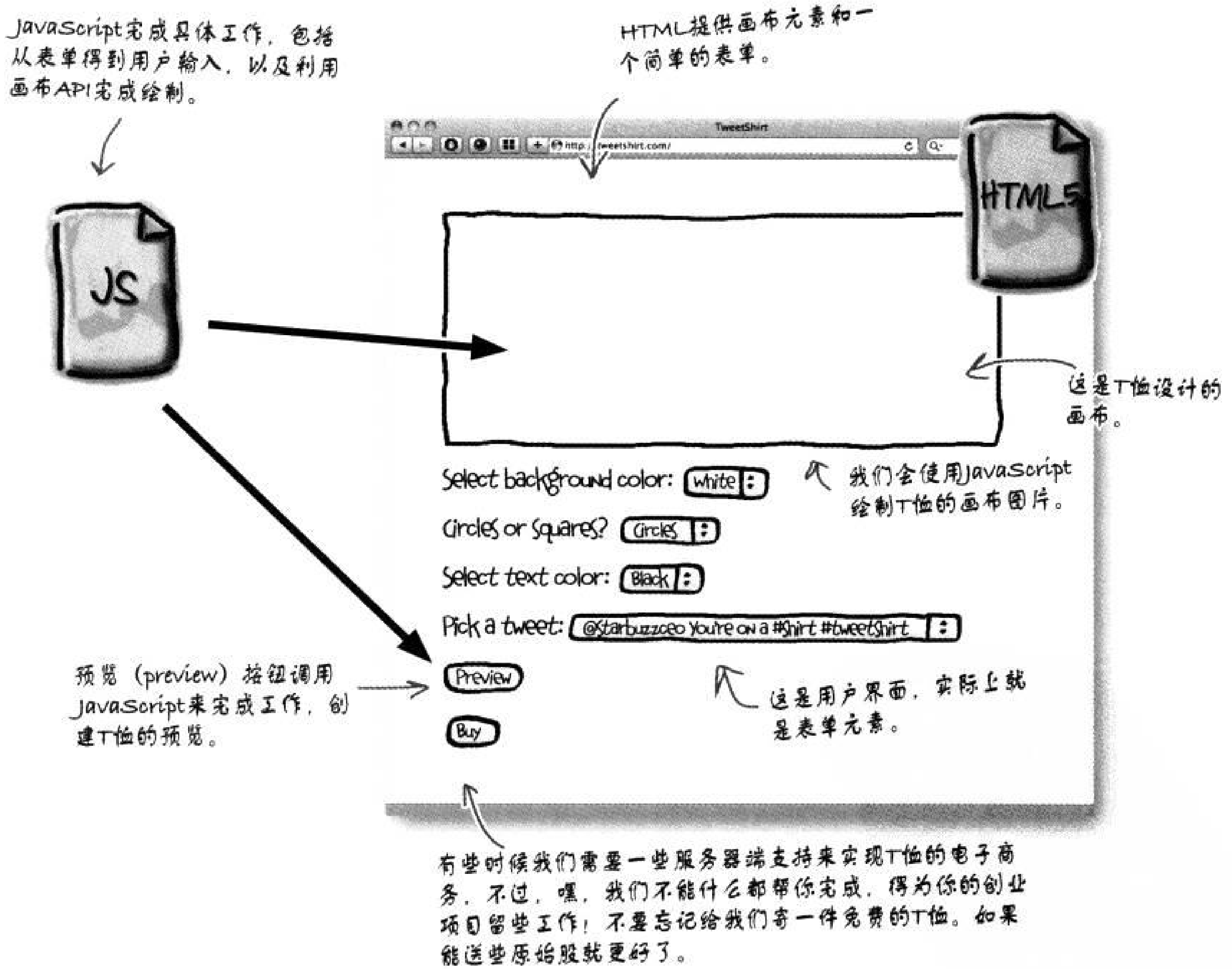
Frank: 我早就不记得了。

Joe: 没关系，我们来一起一步一步回顾一下。首先从顶层的全局视图开始。

TweetShirt：全局视图

开始庞大的实现工作之前，下面先后退一步，来看看这个任务的全局视图。我们要利用一个画布元素构建这个Web应用，同时有一些表单元素作为用户界面，在后台我们要利用JavaScript和画布API完成具体工作。

应该像下面这样：



pin5i.com 拼吾爱

扮演浏览器



下面你会看到T恤界面的表单。你的任务是扮演浏览器，显示这个界面。完成工作后，把你的界面与上一页的界面做个比较，看看你做得对不对。

```
<form>
<p>
    <label for="backgroundColor">Select background color:</label>
    <select id="backgroundColor">
        <option value="white" selected="selected">White</option>
        <option value="black">Black</option>
    </select>
</p>
<p>
    <label for="shape">Circles or squares?</label>
    <select id="shape">
        <option value="none" selected="selected">Neither</option>
        <option value="circles">Circles</option>
        <option value="squares">Squares</option>
    </select>
</p>
<p>
    <label for="foregroundColor">Select text color:</label>
    <select id="foregroundColor">
        <option value="black" selected="selected">Black</option>
        <option value="white">White</option>
    </select>
</p>
<p>
    <label for="tweets">Pick a tweet:</label>
    <select id="tweets">
    </select>
</p>
<p>
    <input type="button" id="previewButton" value="Preview">
</p>
</form>
```

www.pin5i.com 拼吾爱 最新最全的计算机教程 免费下载分享

把你的界面显示在这里。画出Web页面，表单元素放在左边。

再来扮演浏览器

既然已经有了界面，执行这些JavaScript语句，并写出各个界面元素的值。对照本章最后给出的答案检查你做得对不对。

假设你使用这个界面为你的T恤取值。



```
var selectObj = document.getElementById("backgroundColor");
var index = selectObj.selectedIndex;
var bgColor = selectObj[index].value;                ..........

var selectObj = document.getElementById("shape");
var index = selectObj.selectedIndex;
var shape = selectObj[index].value;                  ..........

var selectObj = document.getElementById("foregroundColor");
var index = selectObj.selectedIndex;
var fgColor = selectObj[index].value;                ..........
```

pin51.com 拼吾爱

首先，建立HTML

说得够多了！下面来具体构建这个应用。做其他工作之前，我们只需要一个简单的HTML页面。更新你的index.html文件，如下所示：

```
<!doctype html>
<html lang="en">
<head>
    <title>TweetShirt</title>
    <meta charset="utf-8" />
    <style>
        canvas {border: 1px solid black;}
    </style>
    <script src="tweetshirt.js"></script>
</head>
<body>
    <h1>TweetShirt</h1>
    <canvas width="600" height="200" id="tshirtCanvas">
        <p>Please upgrade your browser to use TweetShirt!</p>
    </canvas>
    <form>

    </form>
</body>
</html>
```

一个符合HTML5的不错的文件，太棒了！

注意，我们把标题改为“TweetShirt”。

下面把所有JavaScript代码放在一个单独的文件中，这样更容易管理。

这里是画布！

我们为使用老浏览器的用户提供了一个消息。

这是表单，包含tweetshirt应用的所有控件。下一页再讨论表单……

BRAIN POWER

如果要把画布上的CSS边框换成用JavaScript在画布上绘制的边框，你还需要了解哪些知识？另外，如果你答得出，说说看你更喜欢哪一种方法（CSS还是JavaScript），为什么？

pin51.com 拼吾爱 最新最全的资源下载站

现在来增加<form>

OK，现在来增加用户界面，这样就可以开始编写一些代码来创建T恤。你以前已经见过这个代码，不过我们增加了一些注解，让代码更清楚一些。输入代码时，一定要参考我们给出的注解。

所有这些代码都放在前一页建立的<form>标记之间。

```
<form>
<p>
    <label for="backgroundColor">Select background color:</label>
    <select id="backgroundColor">
        <option value="white" selected="selected">White</option>
        <option value="black">Black</option>
    </select>
</p>
<p>
    <label for="shape">Circles or squares?</label>
    <select id="shape">
        <option value="none" selected="selected">Neither</option>
        <option value="circles">Circles</option>
        <option value="squares">Squares</option>
    </select>
</p>
<p>
    <label for="foregroundColor">Select text color:</label>
    <select id="foregroundColor">
        <option value="black" selected="selected">Black</option>
        <option value="white">White</option>
    </select>
</p>
<p>
    <label for="tweets">Pick a tweet:</label>
    <select id="tweets">
    </select>
</p>
<p>
    <input type="button" id="previewButton" value="Preview">
</p>
</form>
```

在这里用户可以选择微博T恤设计的背景色。这里的选择有两种：黑色或白色。完全可以增加你自己的颜色。

这里使用另一个选择控件来选择圆或方块，来完成定制设计。用户还可以什么都不选（这会得到一个空白的背景）。

另一个选择控件，用于选择文本的颜色。同样的，这里颜色只能是黑色或白色。

微博放在这里。那么为什么它是空的呢？哈，后面我们会详细说明（提示：我们需要从Twitter得到实际的微博，毕竟这是一个Web应用，对不对？：））。

最后是预览T恤的一个按钮。

如果你熟悉表单，可能会注意到，这个表单居然没有action属性（这说明，点击按钮时它什么也不会做）。稍后我们就会处理这个问题……

pin51.com 拼吾爱 最新最全资源分享

用JavaScript做些计算

标记很不错，不过要由JavaScript来完成这个TweetShirt Web应用的构建。我们要在tweetshirt.js中增加一些代码。现在，我们先做一个小小的工作，只是在T恤上放入随机的方块。不过，即使是做这个简单的工作，首先还需要启用预览按钮，这样当你点击这个按钮时它就会调用一个JavaScript函数。

创建一个tweetshirt.js文件并增加以下代码。

首先得到previewButton元素。

```
window.onload =function() {
    var button = document.getElementById("previewButton");
    button.onclick = previewHandler;
};
```

为这个按钮增加一个点击处理程序，这样当点击这个按钮时（或者在一个移动设备上触摸这个按钮时），就会调用函数previewHandler。

所以，现在点击预览按钮时，就会调用previewHandler函数，我们可以借此机会更新画布，来显示用户设计的T恤。下面先来编写previewHandler：

首先得到画布元素，并要求得到它的2d绘制上下文。

现在需要查看界面中选择了哪个形状。首先，得到id为"shape"的元素。

然后查找选择了哪个元素（方块还是圆），为此得到所选元素的索引，并把它的值赋给变量shape。

如果shape的值是"squares"，就需要画一些方块。画20个怎么样？

```
function previewHandler() {
    var canvas = document.getElementById("tshirtCanvas");
    var context = canvas.getContext("2d");

    var selectObj = document.getElementById("shape");
    var index = selectObj.selectedIndex;
    var shape = selectObj[index].value;

    if (shape == "squares") {
        for (var squares = 0; squares < 20; squares++) {
            drawSquare(canvas, context);
        }
    }
}
```

要绘制各个方块，我们需要利用一个新函数，名为drawSquare，这个函数必须由我们来编写。注意，这里向drawSquare传入了画布和上下文。稍后会看到如何利用这两个参数。

www.cnc51.com 拆吾爱 最新

there are no Dumb Questions

问： selectedIndex是如何工作的？

答： 选择表单控件的selectedIndex属性会返回你在下拉菜单中所选选项的编号。每个选项列表都会转换为一个数组，各个选项会按顺序放在这个数组中。所以，假设你有一个选择列表，包括以下选项："pizza"、"doughnut"和"granola bar"。如果你选择了"doughnut"，selectedIndex就是1（要记住，JavaScript数组索引从0开始）。现在你可能不只是希望得到索引，还想得到这个索引相应的选项值（在这里就是"doughnut"）。要得到这个选项值，首先要使用索引得到数组中的元素，这会返回一个选项对象。得到这个对象的值时可以使用value属性，这会返回选项value属性中的串。



伪代码磁贴

利用你的伪代码能力组织下面的伪代码。我们需要为drawSquare函数编写伪代码。这个函数取一个画布和一个上下文参数，会在画布上绘制一个大小随机的方块。学习后面的内容之前，请对照检查本章最后给出的答案。

我们已经帮你填入了这个磁贴。

```
function drawSquare (          , [上下文] ){



}
```

你的磁贴放在这里！

[画布]

[在位置x,y画一个宽度为w的方块]

"lightblue"是设计初样中方块的颜色。

[为画布中的方块计算一个随机的y位置]

[将fillStyle设置为"lightblue"]

[为方块计算一个随机的宽度]

[为画布中的方块计算一个随机的x位置]

编写drawSquare函数

既然已经完成了最艰巨的任务，明确了伪代码，下面就利用我们现有的知识来编写这个drawSquare函数：

这就是我们的函数，它有两个参数：画布和上下文。

使用Math.random()为方块的宽度和x、y位置创建随机数。后面会做更多说明……

```
function drawSquare(canvas, context) {
    var w = Math.floor(Math.random() * 40);

    var x = Math.floor(Math.random() * canvas.width);

    var y = Math.floor(Math.random() * canvas.height);

    context.fillStyle = "lightblue";
    context.fillRect(x, y, w, w);
}
```

我们选择40作为方块的最大大小，以保证方块不至于过大。

x和y坐标根据画布的宽度和高度确定。我们分别为x和y选择了一个介于0到宽度和0到高度之间的随机数。

这里需要为方块指定一个随机的宽度，以及随机的x和y位置。

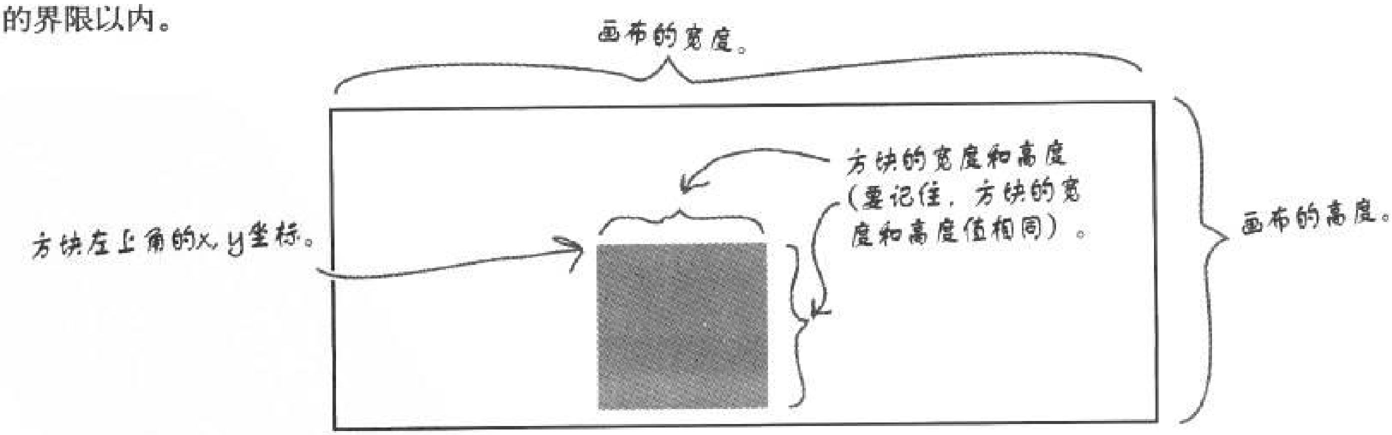
使用fillStyle方法为方块指定一种漂亮的浅蓝色。稍后会更详细地介绍这个方法……

最后，用fillRect画出具体的方块。

《Head First HTML with CSS & XHTML》书中专门有一章介绍颜色，你可以复习一下。

如何确定各个Math.random值乘以哪个数来得到方块的宽度以及x、y位置呢？对于矩形的宽度，我们选择了40，这是因为相对于画布大小这个尺寸很小。由于这是方块，所以高度也使用相同的值。另外，我们选择画布的宽度和高度作为选择x和y值的基础，这样就能保证方块总会落在画布的界限以内。

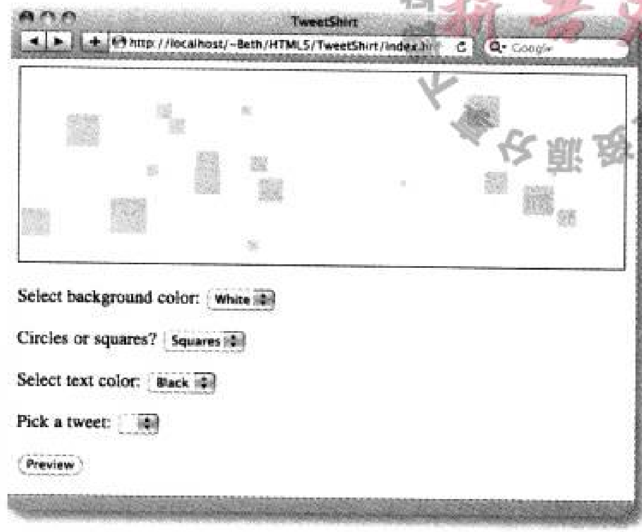
完全可以在你自己的代码中指定其他值（而不是40）！



试一试！

好了，输入所有这些代码之后，现在来运行。在你的浏览器中打开TweetShirt index.html文件。按下预览按钮，你会看到随机的蓝色方块。

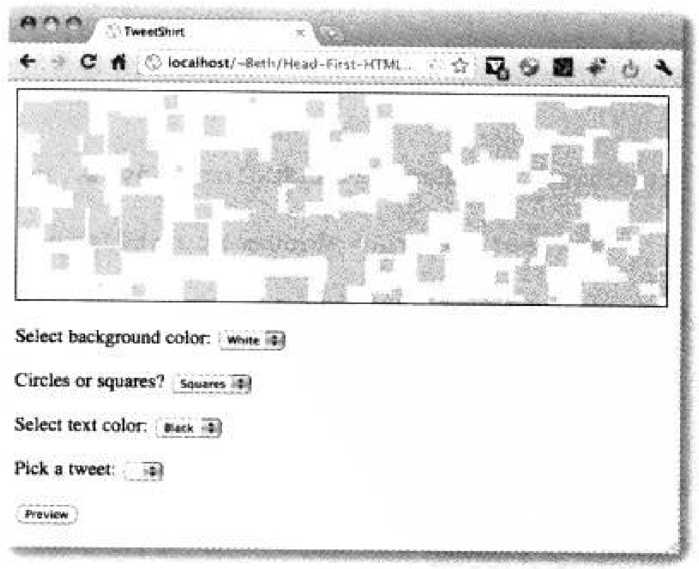
这是我们看到的结果：



不错，这正是我们想看到的！

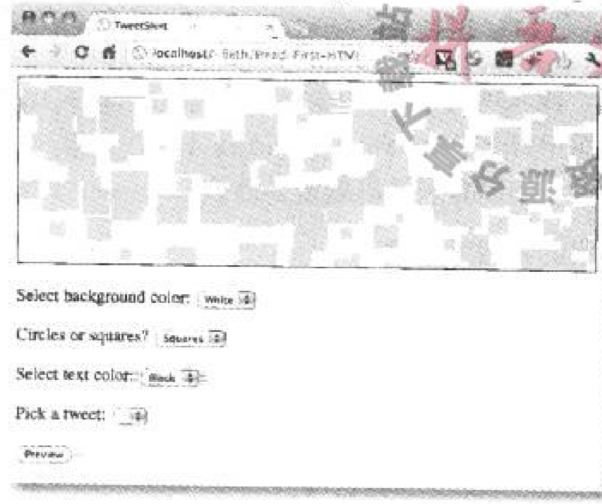


他说得对，我们遇到一个小问题。如果把预览按钮按下多次，就会看到这样的结果。



pin51.com 最新最全 下载资源 拼吾爱

为什么预览时会同时看到老方块和新方块？



这种效果确实很酷……不过这不是我们想要的。我们希望每次按下预览按钮时，新方块会替换原来的老方块（同样的，我们也希望发表微博时，新微博能取代原来的老微博）。

这里的关键是，要记住我们是在画布上绘制像素。按下预览按钮时，你会得到画布，并在上面绘制方块。画布上已有的内容当然会随新像素一同绘制！

不过，不用担心。要修正这个问题，实际上你已经了解所需要的全部知识。我们可以这样做：

1. 从“backgroundColor”选择对象得到所选的背景颜色。

2. 每次开始绘制方块之前，使用fillStyle和fillRect填充画布的背景色。

Sharpen your pencil

为了确保每次点击预览按钮时在画布上只看到新的方块，需要使用用户在“backgroundColor”选择菜单中选择的背景色来填充画布的背景。首先，实现一个函数用这个颜色填充画布。请在下面填空，完成这个代码。学习后面的内容之前先对照检查本章最后给出的答案，看看你做得对不对。

```
function fillBackgroundColor(canvas, context) {

    var selectObj = document.getElementById("________________");

    var index = selectObj.selectedIndex;

    var bgColor = selectObj.options[index].value;

    context.fillStyle = ________________;

    context.fillRect(0, 0, ________________, ________________);

}
```

提示：从选择的选项会得到可以使用的一个颜色字符串，如方块所用的“lightblue”。

提示：我们希望用这种颜色填充整个画布！

增加BackgroundColor调用

已经有了fillBackgroundColor函数，现在只需要确保从previewHandler调用这个函数。首先来增加这个调用，这样在向画布增加其他内容之前才能有一个干净整洁的背景。

```
function previewHandler() {
    var canvas = document.getElementById("tshirtCanvas");
    var context = canvas.getContext("2d");
    fillBackgroundColor(canvas, context);

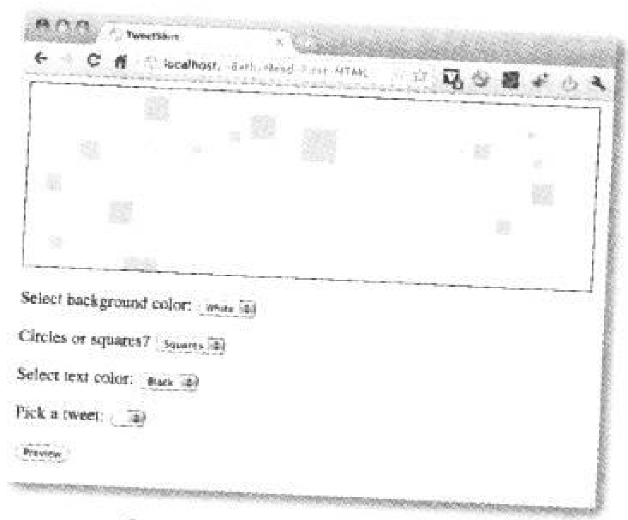
    var selectObj = document.getElementById("shape");
    var index = selectObj.selectedIndex;
    var shape = selectObj[index].value;

    if (shape == "squares") {
        for (var squares = 0; squares < 20; squares++) {
            drawSquare(canvas, context);
        }
    }
}
```

绘制方块之前增加fillBackgroundColor调用，使它覆盖之前绘制的内容，为以后的绘制提供一个干净整洁的背景。

再做一个小测试，确保这个新fillBackgroundColor函数正常工作……

向tweetshirt.js文件增加以上新代码，重新加载浏览器，选择一个背景色，选择方块，再点击预览按钮。然后再次点击。现在每次预览时应该只会看到新方块。



现在每次预览时只会看到新方块。

BRAIN POWER

数一数不同预览中有多少个方块。会不会有时少于20个？有可能。

为什么会这样呢？如何修正这个问题（毕竟，你告诉顾客有20个方块，不希望让他们觉得受骗，对不对）？

pin51.com 拼吾爱 最新最全下载站

JavaScript特写

再来深入介绍fillStyle，因为这是你第一次见到这个属性。fillStyle是上下文的一个属性，保存了在画布上完成绘制时所用的颜色。

类似于fillRect，fillStyle也要通过上下文来控制。

不过，与fillRect不同的是，fillStyle是一个属性，而不是方法，所以需要设置而不是调用。

要把它设置为一个颜色。可以使用CSS中所用的同样的颜色格式。所以可以使用颜色名，如lightblue，或者也可以使用#ccccff或rgb(0, 173, 239)之类的值。你可以试试看！

```
context.fillStyle = "lightblue";
```

注意，与CSS中不同，如果不是使用变量，必须在值两边加上引号。

there are no Dumb Questions

问：我以为要向fillRect传入一个颜色值来设置方块和画布的背景色。我实在不明白fillStyle是怎么做的。它会影响fillRect的工作吗？

答： 问得好。这与你以往的想法可能有一点不同。要记住，上下文是一个对象，可以控制对画布的访问。你使用fillStyle和fillRect时，首先是设置一个属性，告诉画布"不管你接下来画什么，都要用这种颜色"。所以设置fillStyle之后，用颜色填充的任何东西（比如用fillRect）都会使用这种颜色，直到你再将fillStyle设置为另一种不同的颜色来改变颜色。

问：为什么颜色需要有引号，CSS中的属性值就没有？例如，我在设置一个元素的background-color时就没有使用引号。

答： 嗯，CSS与JavaScript是不同的语言，CSS不需要引号。不过，如果在颜色两边未加引号，JavaScript就会认为这个颜色名是一个变量而不是一个字符串，相应地会尝试使用这个变量的值作为颜色。

假设你有一个变量fgColor="black"。可以写出代码：

context.fillstyle=fgColor，这是可以的，因为fgColor的值是"black"。

不过，context.fillStyle=black就不行，因为black不是一个变量（除非你设置了这样一个变量，这可能会带来一些混淆）。你会发现犯了错误，因为你会得到一个JavaScript错误，指出类似这样的消息"无法找到变量：black"（不用担心，我们所有人可能都会犯这个错误，至少一次）。

问：唉，我没辙了。为什么有时候看到的方块数不到20个呢？

答： 方块的x, y和宽度都是随机的。有些方块可能明显，但有些方块就不那么容易看见了。有些方块的x，y位置可能是599，199。所以我们只能看到这个方块的一个像素（因为方块的其余部分都在画布以外）。有些方块可能只有1个像素宽，有些方块的宽度甚至为0，因为Math.random方法可能返回0。或者你也可能生成两个大小和位置完全相同的方块。

不过，对于这个应用，这都属于随机性，所以我们认为这是可以接受的。对于其他应用，就可能需要确保这种情况不要发生了。

pin5i.com 拼吾爱 最新最全资源分享下载站

与此同时，再回到TweetShirt.com……



Jim:我知道，而且居然没用多少代码，真让我吃惊。想想看，如果我们还采用原来的服务器端方法，现在可能还在启动服务器呢。

Frank:看起来现在可以试试在设计中加入圆了。毕竟，它们与方块很像。

Jim:同意。Judy在哪里？她可能知道画圆的API。可能只需要调用一个`fillCircle`方法吧。

Frank:我觉得差不多！别管Judy了，我们自己也能搞定！

pin51.com 拼吾爱



几个小时之后……

Frank: 我真不知道怎么回事。所有代码我都反复检查过了，不过，不管我怎么做，调用fillCircle时，画布上什么都没有。

Judy: 嗯，给我看看你的fillCircle方法。

Frank: 你说“我的方法”，这是什么意思？我没有这个方法呀，我在直接使用画布API中的方法。

Judy: 画布API并没有一个fillCircle方法。

Frank: 啊？我以为有呢，因为我们有一个fillRect...

Judy:嗯，现在知道自以为是的麻烦了吧。来吧，打开浏览器，可以在http://dev.w3.org/html5/2dcontext/找到这个API。

……实际上，画圆不只是调用一个方法那么简单，要比这稍稍复杂一些。你需要先了解路径和弧。

Jim（走进来）: Judy，Frank是不是告诉你我们已经选定圆了？

Frank: 嗯，Jim，enoughway ithway ethay irclecay*!

建议参考piglatin.bavetta.com提供的翻译服务。

* 这是所谓的猪式拉丁语（pig-latin），并非真正的语言，是为了让别人听不懂在讲什么，闹着玩而发明的一种好玩语言。这句话的英文译为“enough for the circle”（不要再提圆了）

pin51.com 拼吾爱 最新最全免费资源下载站

“奇怪地”绘制



具体画圆之前，需要聊一聊路径和弧。先来看路径，画一些三角形。如果想在画布上画一个三角形，并没有一个现成的fillTriangle方法，不过我们确实可以创建三角形。首先创建一个三角形形状的路径，然后用笔划描这个路径，就能在画布上画出一个三角形。

这是什么意思？嗯，假设你想在画布上很精细地画画，可能会拿一支铅笔，在画布上比照着一个模子描出来（暂且把它叫做路径）。你要轻轻地画，使得只有你能看到这个路径。接下来，如果你对这个路径很满意，可以拿一支钢笔（按你选择的粗细和颜色），用这个钢笔描出路径，使每个人都能看到你的三角形（或者你用铅笔描出的任何形状）。

在画布上用线绘出任意的形状时也是这样做的。下面来画一个三角形，看看具体如何实现：

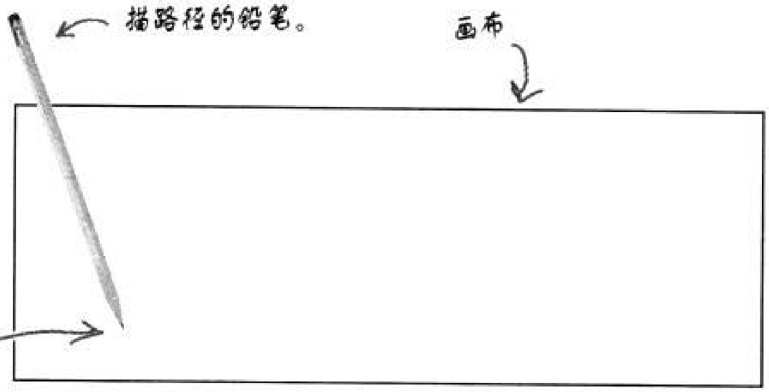
使用beginPath方法告诉画布我们要开始一个新路径。

```
context.beginPath();

context.moveTo(100, 150);
```

用moveTo方法把“铅笔”移动到画布上的一个指定点。可以认为把铅笔放在这一点上。

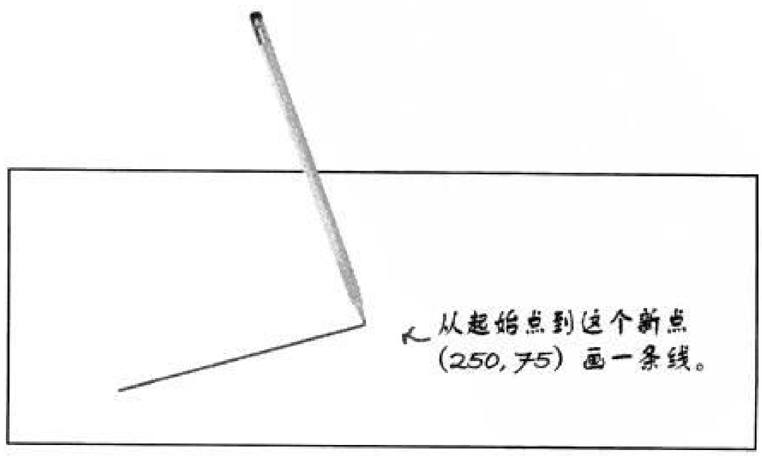
这里我们把铅笔放在x = 100，y = 150的位置上。这是路径的第一个点。



lineTo方法描路径，从铅笔的当前位置描到画布上的另一个点。

```
context.lineTo(250, 75);
```

铅笔位于100，150，这里我们把路径从这个点延伸到下一个点（x=250，y=75）。

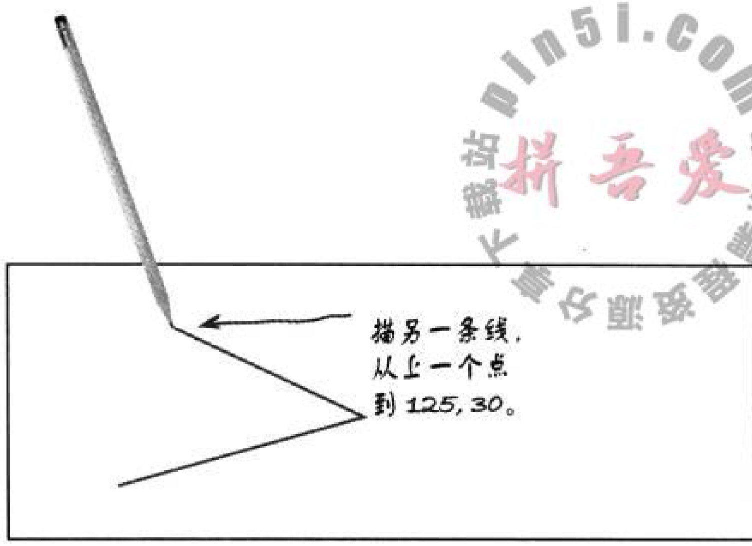


pin51.com 拼吾爱

我们已经完成了三角形的第一条边，现在再来画第二条边：

```
context.lineTo(125, 30);
```

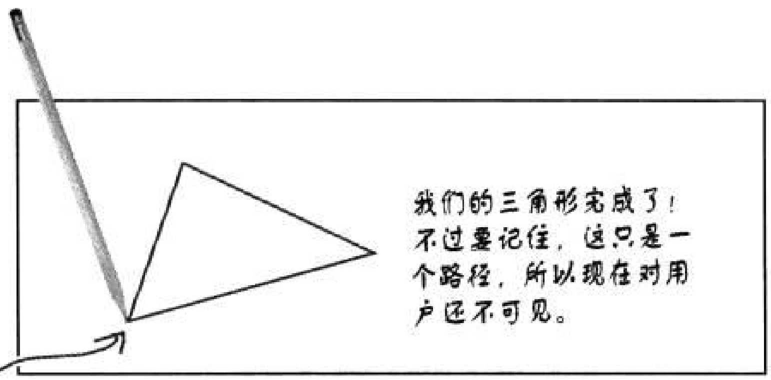
这里从铅笔的当前位置(250, 75)描到一个新位置 x = 125, y = 30。这样就完成了第二条线。



就快完成了！现在只需要再描一条线就可以完成这个三角形了。为此，只需要用closePath方法闭合这个路径。

```
context.closePath();
```

closePath方法将路径的起始点(100, 150)连接到当前路径的最后一个点(125, 30)。





Exercise

现在已经有了路径！然后呢？

当然要使用这个路径来画线，并用颜色填充你的形状！下面创建一个简单的HTML5页面，其中包含一个画布元素，输入前面的所有代码。再运行试试看。

```
context.beginPath();
context.moveTo(100, 150);
context.lineTo(250, 75);
context.lineTo(125, 30);
context.closePath();
```

这是目前为止的代码。

```
context.lineWidth = 5;                .............................

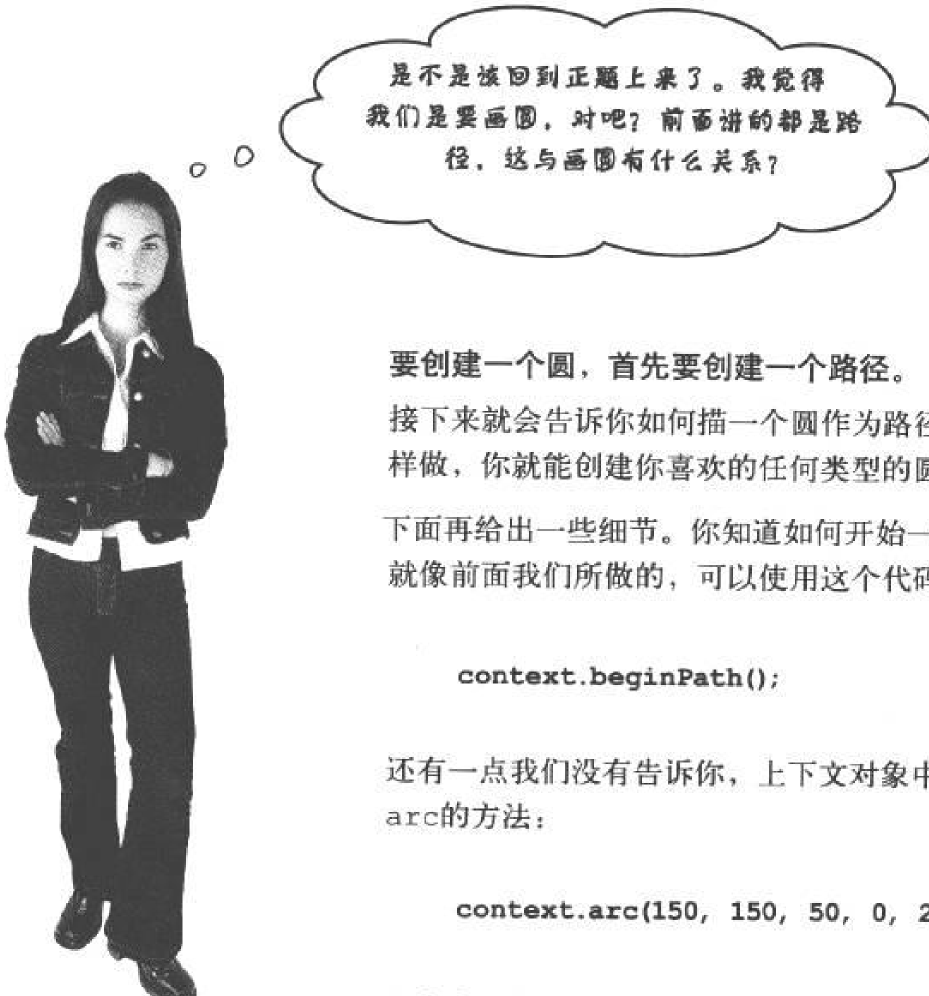
context.stroke();                     .............................

context.fillStyle = "red";            .............................

context.fill();                       .............................
```

这里有一些新代码。给这些代码加上注解，指出你认为这些代码会做什么。加载页面，你的想法对吗？请对照检查本章最后给出的答案。

pin5i.com 拼吾爱 最新最全资源 下载站



要创建一个圆，首先要创建一个路径。

接下来就会告诉你如何描一个圆作为路径。一旦知道了怎样做，你就能创建你喜欢的任何类型的圆了。

下面再给出一些细节。你知道如何开始一个路径，对不对？就像前面我们所做的，可以使用这个代码：

```
context.beginPath();
```

还有一点我们没有告诉你，上下文对象中还有一个名为arc的方法：

```
context.arc(150, 150, 50, 0, 2 * Math.PI, true);
```

你是不是还记得几何课我们学过的周长 = 2πR？现在要把它记住……

这个方法做什么？哈，我们会在下一页找到有关的细节。不过，你可能会猜到，这会沿一个圆描出路径。

pin51.com 拼吾爱

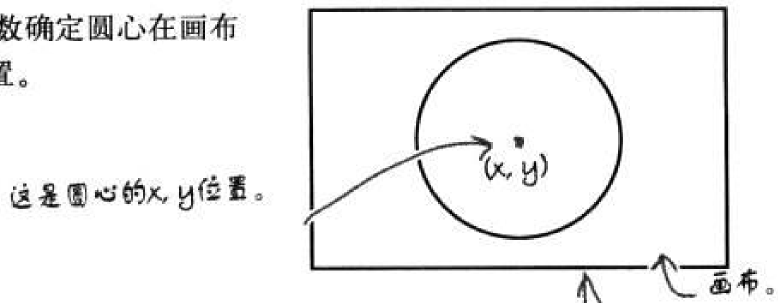
分解arc方法

下面深入分析arc方法，查看它的各个参数：

```
context.arc(x, y, radius, startAngle, endAngle, direction)
```

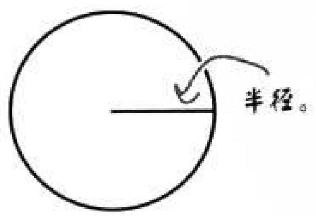
arc方法就是要指定如何沿一个圆描出路径。下面来看各个参数的作用：

x,y x和y参数确定圆心在画布上的位置。




```
context.arc(x, y, radius,
```

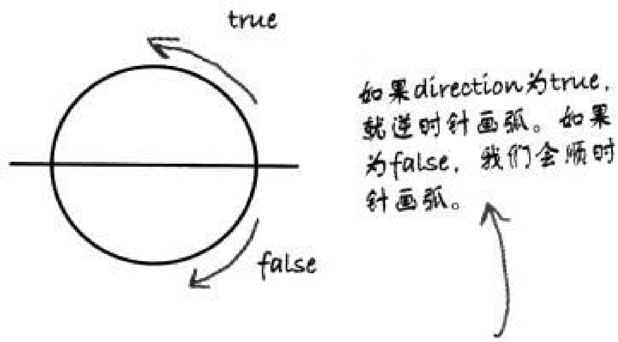
radius 这个参数用来指定圆的半径（宽度的1/2）。





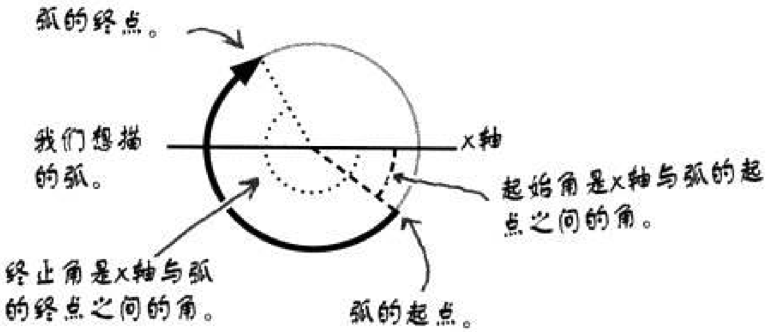
pin51.com 拼吾爱

direction 确定以逆时针方向还是顺时针方向创建圆弧路径。如果方向为true，就是逆时针；否则如果为false，就是顺时针。



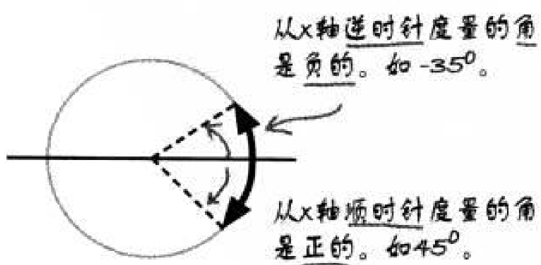

```
startAngle, endAngle, direction)
```

startAngle, endAngle 弧的起始角和终止角确定了路径在圆上的起点和终点。



下面是重点！

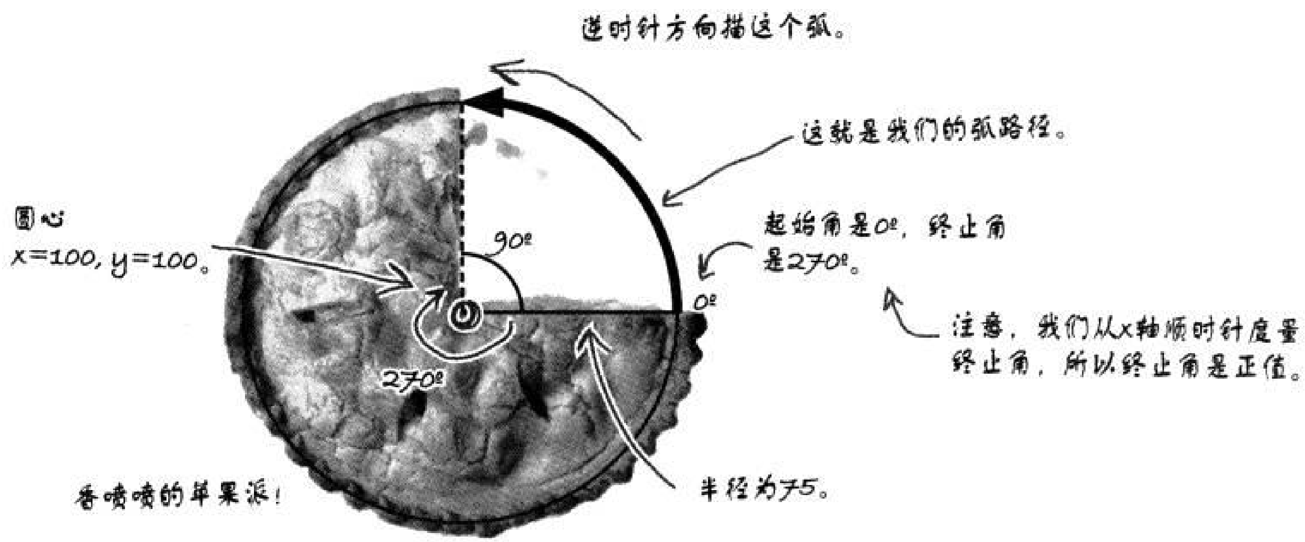
千万别跳过不看。角可以按负方向（从x轴逆时针）度量，也可以按正方向（从x轴顺时针）度量。这与arc路径的方向参数不同（下一页你就会看到）。



pin51.com 拼吾爱 最新最全资源分享下载站

浅尝弧的使用

现在我们需要一个好例子。假设你希望在一个圆心位于x=100,y=100的圆上描一个弧，而且这个圆的宽度为150像素（或者半径为75像素）。另外，你想描的路径只是这个圆的1/4，如下：



现在来创建一个arc方法调用，描出这个路径：

1. 从圆心x, y点开始：100, 100。

```
context.arc(100, 100, __, __, ______________________, ____);
```

2. 接下来，需要这个圆的半径，75。

```
context.arc(100, 100, 75, __, ______________________, ____);
```

3. 起始角和终止角呢？嗯，起始角为0，因为起点位于相对于x轴的0°角。终止角是x轴与弧的终点之间的角。由于我们的弧是一个90°的弧，所以终止角为270°（90+270=360）（注意，如果按负值或逆时针方向来度量，那么终止角就是-90°）。

```
context.arc(100, 100, 75, 0, degreesToRadians(270), ____);
```

稍后还会再来介绍这个方法。它的作用就是将（我们习惯的）度转换为弧度（上下文对象更愿意使用弧度）。

4. 最后，按逆时针方向描这个弧，所以使用true。

```
context.arc(100, 100, 75, 0, degreesToRadians(270), true);
```

pin5i.com 拼吾爱

我说度，你却说弧度

我们每天都在谈论圆的角度："漂亮的360°"，或者"我沿着那条路走，然后做了一个180°大转弯"，或者……嗯，应该能想到，还有很多很多。唯一的问题是，我们是从度来考虑，而画布上下文却从弧度考虑。

弧度只是角度的另一种度量。1弧度就等于180/3.14159265……（或者180/π）。



现在可以告诉你：

    360 度 = 2Pi 弧度

如果你想从现在开始自己在头脑里计算弧度，那你可得费心了。或者，如果出于某些原因你不想自己来做这个计算，可以给你一个方便的函数帮你完成这个工作：

```
function degreesToRadians(degrees) {
    return (degrees * Math.PI)/180;
}
```

如果你想从度来考虑，但是要得到弧度来绘制一个弧，可以使用这个函数。

你可能在地理定位一章中见到过这个函数。

要从度得到弧度，需要乘以π再除以180。

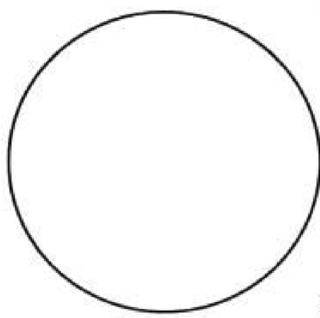
313页上，你已经见到我们使用2 × Math.PI来指定一个圆的终止角。你也可以这样做……或者直接使用degreesToRadians(360)。

扮演浏览器



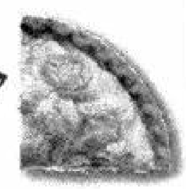
解释这个arc方法调用，在图上画出所有值，包括这个方法创建的路径。

```
context.arc(100, 100, 75, degreesToRadians(270), 0, true);
```



在这个圆上标出所有参数，然后画出这个方法调用创建的路径。

提示：吃掉这部分后，这个派还剩下什么？



pin51.com 拼吾爱

再来编写TweetShirt的圆代码

既然知道了如何画圆，现在再回到TweetShirt，增加一个新函数drawCircle。我们想画20个随机的圆，就像前面的方块一样。要画这些圆，首先我们需要确定用户在形状菜单中选择了圆。下面把这个代码增加到previewHandler函数。

编辑tweetshirt.js文件，并增加下面的新代码。

```
function previewHandler() {
    var canvas = document.getElementById("tshirtCanvas");
    var context = canvas.getContext("2d");
    fillBackgroundColor(canvas, context);

    var selectObj = document.getElementById("shape");
    var index = selectObj.selectedIndex;
    var shape = selectObj[index].value;

    if (shape == "squares") {
        for (var squares = 0; squares < 20; squares++) {
            drawSquare(canvas, context);
        }
    } else if (shape == "circles") {
        for (var circles = 0; circles < 20; circles++) {
            drawCircle(canvas, context);
        }
    }
}
```

这个代码看上去与测试方块的代码几乎完全相同。如果用户选择了圆而不是方块，我们就会用drawCircle函数画20个圆（现在就来编写这个函数）。

将画布和上下文传入drawCircle函数，就像调用drawSquares函数时一样。

BRAIN POWER

要画一个完整的圆，起始角和终止角是什么？

你要使用什么方向：逆时针还是顺时针？这会有影响吗？

答：画圆时，起始角为0°，终止角为360°。使用什么方向没有任何影响，因为你要画一个完整的圆。

编写drawCircle函数……

现在来编写drawCircle函数。要记住，我们只需要画一个随机的圆。其他代码就是将这个函数调用20次。

```
function drawCircle(canvas, context) {
    var radius =  Math.floor(Math.random() * 40);
    var x = Math.floor(Math.random() * canvas.width);
    var y = Math.floor(Math.random() * canvas.height);


    context.beginPath();
    context.arc(x, y, radius, 0, degreesToRadians(360), true);

    context.fillStyle = "lightblue";
    context.fill();
}
```

就像画方块一样，这里使用40作为最大半径，来避免圆变得过大。

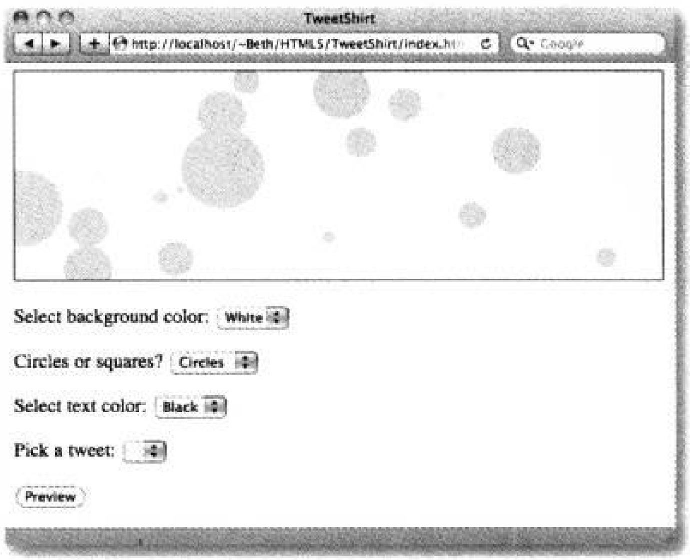
同样的，圆心的x和y坐标根据画布的宽度和高度确定。我们分别选择了介于0与宽度和高度之间的一个随机数。

这里使用终止角360°来得到一个完整的圆。可以沿圆逆时针绘制，不过对于一个圆，使用哪个方向并不重要。

这里还是使用"lightblue"作为fillStyle，然后用context.fill()填充这个路径。

…… 试一试！

现在输入这个代码（别忘了还要增加degreesToRadians函数），保存，然后在你的浏览器加载这个文件。我们会看到下面的结果（由于这些是随机的圆，你看到的可能稍有不同）：



中场休息





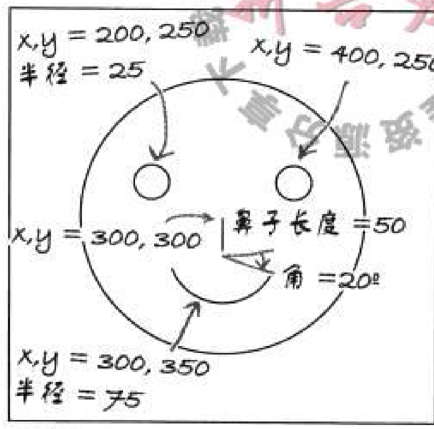
茶点时间

哇！刚才这几页真有意思。虽然我们还不相识，不过我们准备了一些茶点。不妨稍稍休息一会，吃点茶点？但不要以为在你吃茶点时会很无聊，我们还会给你一些有趣的东西（请看右边的练习）。

来，坐下来，休息一会，慢慢吃，给你的大脑和肚子都装点东西。然后再回来完成TweetShirt代码！

www.aibbt.com 让未来触手可及

在右边我们会看到一个笑脸（如果你愿意，也可以是一个巧克力笑脸饼干）。下面的代码就是用来画这个笑脸的，已经快完成了，我们需要你的帮助来完成它。经过这一章的学习，加上你之前的努力，你已经掌握了所需的全部知识，完全可以完成这个任务。完成之后，可以对照检查本章最后给出的答案。



这就是我们想要的。不过你可能想做一些真正的巧克力笑脸饼干……

```
function drawSmileyFace() {
    var canvas = document.getElementById("smiley");
    var context = canvas.getContext("2d");

    context.beginPath();
    context.arc(300, 300, 200, 0, degreesToRadians(360), true);
    context.fillStyle = "#ffffcc";
    context.fill();
    context.stroke();

    context.beginPath();
    context.arc(____, ____, 25, ____, __________, true);
    context.stroke();

    context.beginPath();
    context.arc(400, ____, ____, ____, _________, _____);
    context.stroke();

    context.beginPath();
    context.________(____, ____);
    context.________(____, ____);
    context._________();

    context.beginPath();
    context.____(300, 350, ____, degreesToRadians(____), degreesToRadians(____), _____);
    context.stroke();
}
```

圆脸。这部分我们帮你完成了。注意这个圆用黄色填充。

这是左眼。

这是右眼。

这是鼻子。

这里是嘴。这是最难的一部分！

pin51.com 拼吾爱

欢迎回来……

休息过，放松过，现在你又回来了，我们已经到了这个创业项目的最后冲刺阶段。查看之前所做的全部工作，现在只剩下一件事，就是在画布预览中显示微博和其他文本。



要在画布上放一个微博，首先需要你的一些最新的微博，可以从中选择，我们将使用JSONP来完成这个任务。如果你还记得第6章的内容，应该知道如何来做到。如果有必要，可以返回第6章简单地复习一下。我们要做到：

1. 在tweetshirt.html文件的最后增加一个<script>，调用Twitter JSONP API。我们要请求一个指定用户的最新的状态更新。

2. 实现一个回调来得到Twitter发回的微博。我们将在第1步增加的<script>的URL中使用这个回调的函数名。

这是TweetShirt的HTML文件。

假设这里是head元素，另外form元素在这里（我们想少占些篇幅，少用点纸，这样就能少砍些树）。

这是我们的JSONP调用：它通过调用Twitter URL获取所创建的JSON，然后将这个JSON传递到回调函数（稍后会定义这个回调）。

这是Twitter API调用。我们请求一个最近更新，这会提供用户最新的状态。

将这个改为你的用户名，或者如果愿意，也可以改为任何其他用户名。

```
<html>
...
<body>
    <form>
    ...
    </form>
    <script src="http://twitter.com/statuses/user_timeline/wickedsmartly.json?
callback=updateTweets">
    </script>
</body>
</html>
```

这是回调函数，JSON将传入这个函数。

把这个代码输入到文本文件的一行上（这行代码太长，无法在书中用一行显示）。

这里的内容很多。如果有些不太清楚，一定要返回到第6章复习一下JSONP是如何工作的。

www.pin5i.com 拼吾爱 最新资源分享下载站

得到微博

我们已经完成了最难的部分，从Twitter得到微博。现在需要把它们增加到页面<form>中的tweets<select>元素。再复习一下：调用回调函数时（这里就是函数updateTweets），Twitter会返回一个响应，其中包含JSON格式的微博。

Twitter的响应是一个微博数组。每个微博包含大量数据，我们要用到的只是微博的文本。

编辑你的tweetshirt.js文件，在最下面增加这个updateTweets函数。代码如下：

这是我们的回调。

传入一个响应，其中包含最近更新中的微博（作为一个微博数组）。

得到表单中tweets选择元素的一个引用。

```
function updateTweets(tweets) {
    var tweetsSelection = document.getElementById("tweets");

    for (var i = 0; i < tweets.length; i++) {
        tweet = tweets[i];
        var option = document.createElement("option");
        option.text = tweet.text;
        option.value = tweet.text.replace("\"", "'");

        tweetsSelection.options.add(option);
    }

    tweetsSelection.selectedIndex = 0;
}
```

对于微博数组中的每一个微博，我们要：

从数组得到一个微博。

创建一个新的选项元素。

将其文本设置为这个微博。

将它的值也设置为同样的文本，不过我们会对这个串稍做处理，把双引号换成单引号（这样可以避免HTML中的格式化问题）。

然后取这个新选项，把它增加到表单中的tweet选择元素。

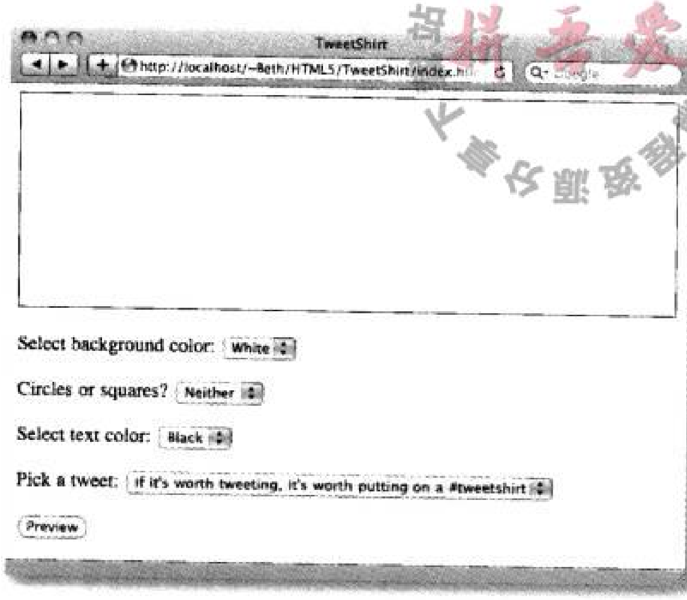
最后，将<select>的selectedIndex设置为0（其中包含的第一个选项元素），确保选中第一个微博。

对各个微博完成这个处理后，就有了一个<select>元素，其中对应每个微博有一个选项。

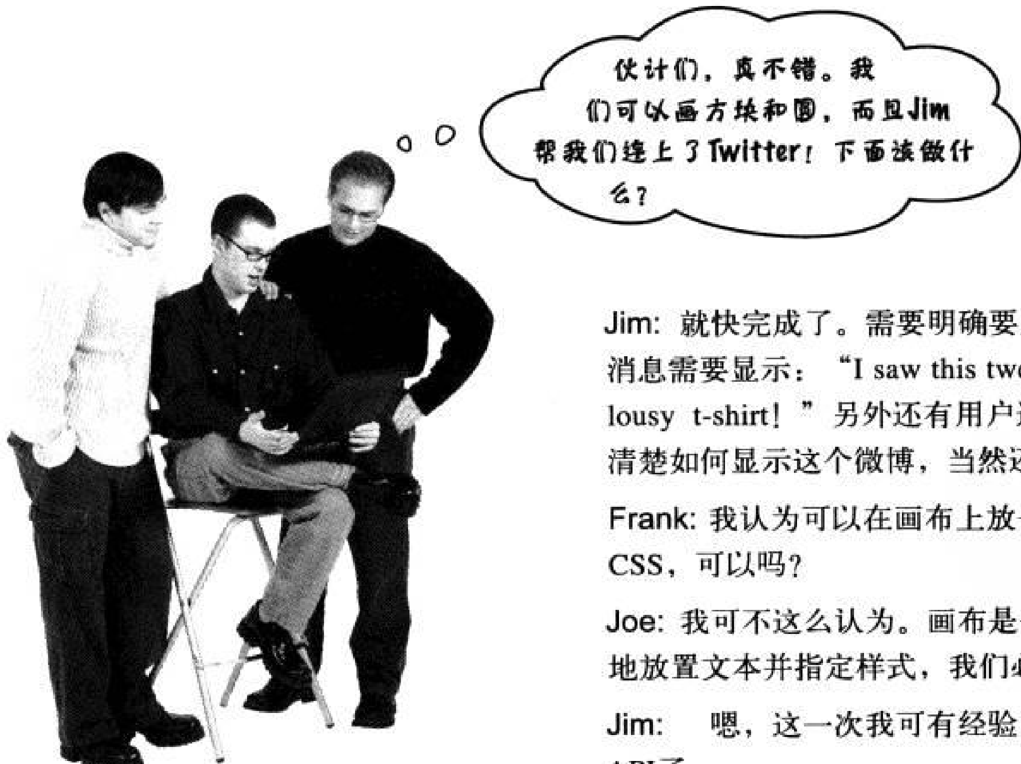
pin51.com 拼吾爱 最新最全 资源分享下载站

试一试微博

来做一个简单的测试。确保已经为`tweetshirt.js`和`index.html`增加了所有代码。另外要保证使用一个Twitter用户名，对应这个用户名的script src URL中有最新的微博（这样可以确保你总能看到一些微博）。加载这个页面，点击微博选择元素。你会看到：



这里是微博菜单，其中包含真正的微博。太酷了！



Frank的设计

Jim: 就快完成了。需要明确要显示的所有文本。我们有两个消息需要显示："I saw this tweet"和"and all I got was this lousy t-shirt!"另外还有用户选择显示的微博。现在必须搞清楚如何显示这个微博，当然还要对文本应用一些样式。

Frank: 我认为可以在画布上放一些文本，然后对它应用一些CSS，可以吗？

Joe: 我可不这么认为。画布是一个绘制区，我觉得不能简单地放置文本并指定样式，我们必须在画布上绘制文本。

Jim: 嗯，这一次我可有经验了，我已经在查看文本的相关API了。

Joe: 很好，我还没看过呢；看上去怎么样呢？

Jim: 还记得arc方法吗？我们必须使用这个方法定制绘制所有文本。

拼吾爱

Frank: 你在开玩笑吧？如果是这样，我想这个周末算是泡汤了。

Jim: 哈哈，理解！不过我说着玩呢！实际上有一个fillText方法，需要指定将在画布上绘制的文本，以及要在哪里绘制（文本的x,y位置）。

Joe: 听上去很简单啊。那么样式的差别如何体现呢？我记得“初样”中微博文本是斜体，其余的文本是粗体。

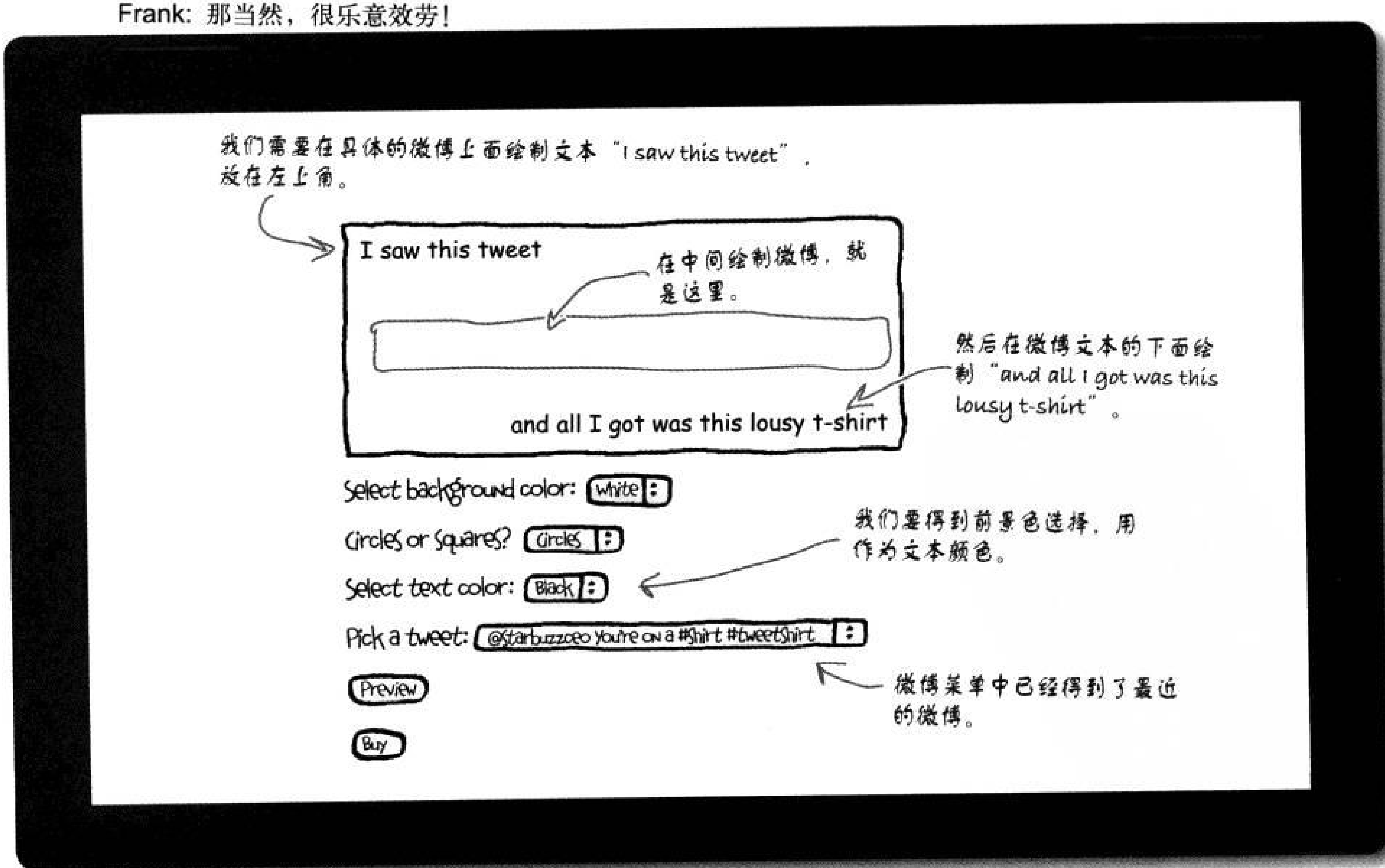
Jim: 需要再多做一些了解，有很多不同的方法来设置对齐方式、字体和填充样式，不过我还不太清楚如何使用这些方法。

Frank: 也许我能帮忙，不过怎么没有CSS？

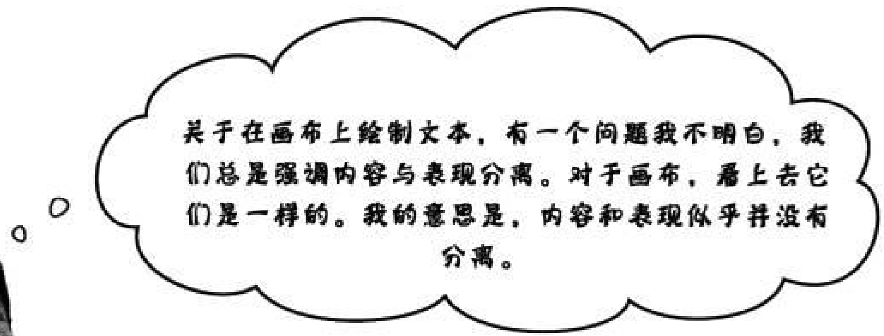
Jim: 抱歉，就像Joe说的，这是一个在画布上完成绘制的API，它不会以任何方式使用HTML或CSS。

Joe: 嗯，来看看这个API吧，然后我们就能试着在画布上绘制文本“I saw a tweet”。来吧，Frank，一起来，情况没那么糟，相信我们能用到你掌握的那些关于字体和样式的知识。

Frank: 那当然，很乐意效劳！



pin51.com 拼吾爱 最新最全资源分享下载站





这个问题问得好。

现在来研究这什么要这样做。记住，设计画布是为了提供一种方法在浏览器中表示图片。画布中的所有一切都认为是表现，而不是内容。所以尽管你通常认为文本（当然还有微博）是内容，但在这里，必须把它认为是表现。这是设计的一部分。就像艺术家把文字作为艺术创作的一部分一样，现在你就在使用微博作为T恤设计创作的一部分。

表现和内容分离是一个很好的想法，一个主要原因在于，这样一来，浏览器可以聪明地处理不同的情况下如何表现内容；例如，一个新闻网站的文章在大屏幕上会用一种方式表现，而在你的手机上则会以一种完全不同的方式表现。

对于我们的T恤设计，我们希望画布中的内容更像一个图像：不管如何浏览都应当采用同样的方式显示。

所以，下面开始在画布上绘制文本，让这个创业项目运转起来吧！

拼吾爱



代码磁贴

这是你第一次尝试画布文本。下面我们给出了drawText的部分代码，我们将调用这个方法在预览画布上绘制所有文本。看看你能不能完成这个代码，在画布上绘制“I saw this tweet”和“and all I got was this lousy t-shirt”，绘制具体微博的代码以后再完成。学习后面的内容之前先对照检查本章最后给出的答案，看看你的答案对不对。

```
function drawText(canvas, context) {

    var selectObj = document.getElementById("________________");

    var index = selectObj.selectedIndex;

    var fgColor = selectObj[index].value;

    context.______________ = fgColor;

    context.___________ = "bold 1em sans-serif";

    context._________________ = "left";

    context.___________(_"_____________________, 20, 40);


    // Get the selected tweet from the tweets menu
    // Draw the tweet


    context.font = "_______________________";

    context.______________ = "__________";

    context.____________("and all I got was this lousy t-shirt!",

            ______________________, ______________________);

}
```

提示：这是“I saw this tweet”文本的x,y位置。

目前，我们只是在这里给出注释，将来会在这里放入绘制微博文本的具体代码。

提示：微博将用一种斜体serif字体，不过我们希望这个文本使用粗体sans-serif字体。

提示：我们希望这个文本放在右下角。

我们希望在距画布右边20像素、距底边40像素的位置绘制这个文本，以便与第一行文本平衡。

`foregroundColor` `fillStyle` `fillText` `bold 1em sans-serif` `fillText` `font` `textAlign` `canvas.width-20` `right` `textAlign` `canvas.height-40` `fillCircle` `fillRect` `"I saw this tweet"` `left`

pin51.com 拼吾爱

画布特写

既然有机会在画布上绘制你的第一个文本，现在可以更仔细地分析画布API中提供的文本方法和属性。在练习中你已经看到，这是一个相当底层的API，你必须告诉上下文你要绘制什么文本、要使用哪个位置，另外要使用什么字体。

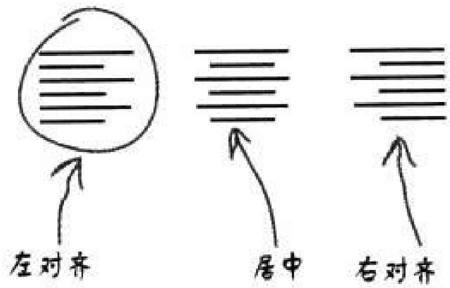
在这个“特写”中，我们将分析对齐方式、字体和基线属性，另外还会详细分析填充和笔划方法，等你翻过这一页，你就会成为名符其实的画布文本专家！

对齐

`textAlign`属性指定了文本的锚点位置。默认值为“start”。

```
context.textAlign = "left";
```

可取值包括：start、end、left、right和center。在从左到右的语言中，Start和end与left和right含义相同，比如英语，但在从右到左的语言（如希伯来语）中则正好相反。



填充和笔划

就像矩形一样，我们也可以用笔划和填充方式绘制文本。需要提供要绘制的文本、x, y位置和一个可选的参数maxwidth，如果文本宽度大于maxwidth，这会导致文本相应缩放。

```
context.fillText("Dog", 100, 100, 200);

context.strokeText("Cat", 100, 150, 200);
```



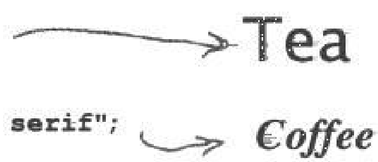
如果文本宽度大于200，它会自动缩放以适应。

拼吾爱

字体

要设置字体属性，可以使用CSS中使用的相同格式，这很方便。如果要指定每一个属性值，按顺序则包括：字体样式、粗细、大小和字体族。

```
context.font = "2em Lucida Grande";
context.fillText("Tea", 100, 100);
context.font = "italic bold 1.5em Times, serif";
context.fillText("Coffee", 100, 150);
```



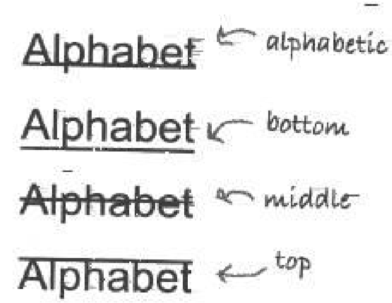
规范建议只使用矢量字体（位图字体可能不能很好地显示）。



基线

textBaseline属性设置字体中的对齐点，并确定字母所在的基线。要看这个基线对文本有什么影响，可以在绘制文本所在的x,y点上画一条线。

```
context.beginPath();
context.moveTo(100, 100);
context.lineTo(250, 100);
context.stroke();
context.textBaseline = "middle";
context.fillText("Alphabet", 100, 100);
```



可取值包括：top、hanging、middle、alphabetic、ideographic和bottom。默认值是alphabetic。可以尝试不同的值来看你需要哪一种（更多细节可以参考规范）。

pin5i.com 拼吾爱

试一试drawText

既然已经对这个API有了更多的了解，就不要只是纸上谈兵了，输入你在代码磁贴练习中创建的代码。这就是你的代码，我们已经把磁贴变成了具体代码：

```
function drawText(canvas, context) {
    var selectObj = document.getElementById("foregroundColor");
    var index = selectObj.selectedIndex;
    var fgColor = selectObj[index].value;

    context.fillStyle = fgColor;
    context.font = "bold 1em sans-serif";
    context.textAlign = "left";
    context.fillText("I saw this tweet", 20, 40);



    context.font = "bold 1em sans-serif";
    context.textAlign = "right";
    context.fillText("and all I got was this lousy t-shirt!",
        canvas.width-20, canvas.height-40);
}
```

稍后我们会在这里补充绘制微博文本的代码……

输入这些代码后，更新你的`previewHandler`函数来调用`drawText`函数，在浏览器中加载这个代码，来试一试。你应该能和我们一样看到这样的结果：

这里是文本。我们在正确的位置上显示了sans-serif字体的粗体文本。



而且下面的文本右对齐。

Sharpen your pencil

试着完成drawText函数。你要得到选择的微博，将字体设置为一种斜体serif字体，比默认大小稍大一点(1.2em)，确保文本左对齐，另外位置在x = 30, y = 100。完成这最后一步之后我们就能看到TweetShirt了！

把你的代码写在上面，不要偷看下一页的答案（说真的）！

最新最全最新 拼吾爱 下载资源分享站 pin5i.com

完成drawText函数

下面给出我们的答案。与你的代码有不同吗？如果你还没有输入你的代码，请输入下面的代码（或者如果你愿意，也可以输入你自己的版本），然后重新加载你的index.html。下一页会给出我们的测试结果。

```
function drawText(canvas, context) {
    var selectObj = document.getElementById("foregroundColor");
    var index = selectObj.selectedIndex;
    var fgColor = selectObj[index].value;

    context.fillStyle = fgColor;
    context.font = "bold 1em sans-serif";
    context.textAlign = "left";
    context.fillText("I saw this tweet", 20, 40);

    selectObj = document.getElementById("tweets");
    index = selectObj.selectedIndex;
    var tweet = selectObj[index].value;
    context.font = "italic 1.2em serif";
    context.fillText(tweet, 30, 100);

    context.font = "bold 1em sans-serif";
    context.textAlign = "right";
    context.fillText("and all I got was this lousy t-shirt!",
        canvas.width-20, canvas.height-40);
}
```

你不需要再将微博文本左对齐，因为这里已经设置了对齐方式。

从微博菜单获取所选的选项。

字体设置为斜体serif，稍稍大一点……

……并在位置30, 100绘制这个文本。



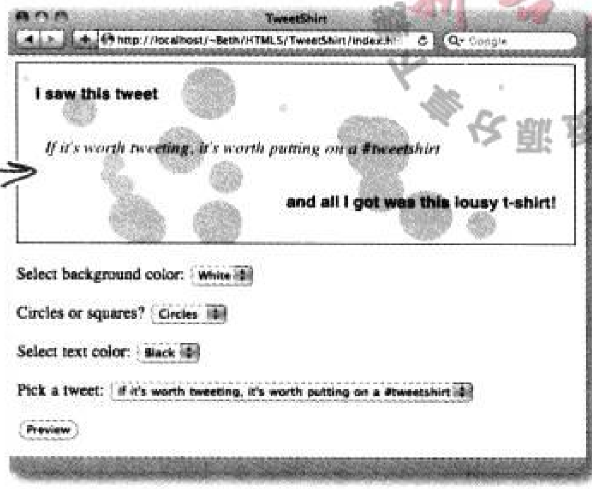
pin51.com 拼吾爱 最新最全的资源分享站

一个简单测试然后~~去进午餐~~启动吧！

希望你与我们看到的一样！不错吧，是不是？再对这个界面做些真正的质量保证检验：试试所有颜色和形状的组合，或者换个你希望的其他用户名。

是不是准备真正启动这个项目了？那就开始吧！

T恤预览中显示了微博。太棒了！





还记得这位TweetShirt创始人吗？

www.pin51.com 拼吾爱 最新最全 下载站 分享资源



1 首先需要一个图像。我们已经在TweetShirt文件夹里放了一个名为`twitterBird.png`的图像。要把这个图像放在画布上，首先需要一个JavaScript image对象。可以这样来得到：

```
var twitterBird = new Image();
twitterBird.src = "twitterBird.png";
```

创建一个新的image对象。

将它的源设置为Twitter小鸟图像。

2 现在下一部分就很自然了，要用一个上下文方法在画布上绘制这个图像，这个方法你应该能猜到，名为`drawImage`。

```
context.drawImage(twitterBird, 20, 120, 70, 70);
```

使用drawImage方法。

这是我们的image对象。

指定x, y位置、宽度和高度。

3 关于图像还有一点需要知道：图像并不总会立即加载，所以在绘制图像之前需要确保图像已经完全加载。那么采取行动之前如何等待某个东西完全加载呢？没错，要实现一个onload处理程序：

```
twitterBird.onload =function() {
    context.drawImage(twitterBird, 20, 120, 70, 70);
};
```

这里指出：图像加载时，就执行这个函数。

使用上下文的drawImage方法在画布上绘制图像。

pin51.com 拼吾爱

Exercise

看看你能不能利用Judy给出的代码完成drawBird函数。drawBird函数取一个画布和上下文参数，在画布上绘制一个小鸟。假设利用这个函数我们将把“twitterBird.png”放在x=20， y=120位置上，宽度和高度都为70。我们已经帮你写出了这个方法的声明和第一行代码。本章最后会给出答案。

```
function drawBird(canvas, context) {
    var twitterBird = new Image();




}
```

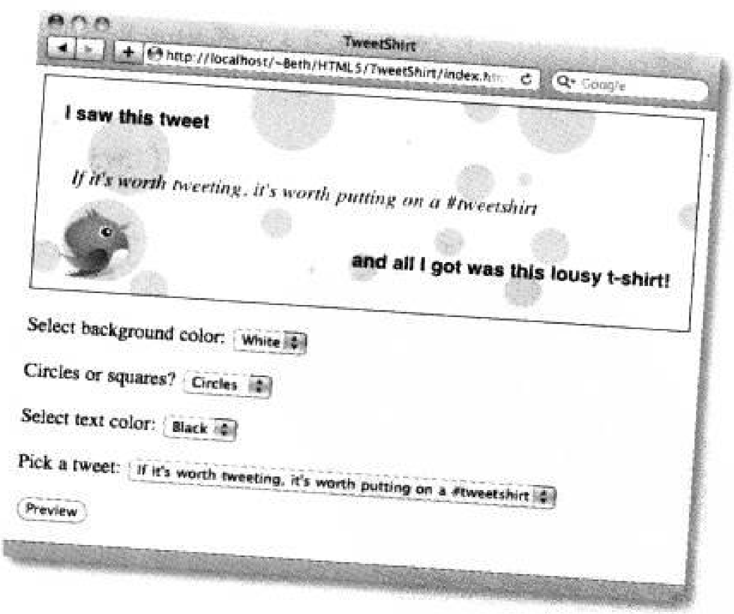
你的代码写在这里。 →

别忘了要在previewHandler函数中增加一个drawBird函数调用。

还有一个测试

反复检查你的代码，再做一个测试！哇，现在看起来真是太完美了。

用圆或方块多试几次。你会注意到，我们使用了一个有透明背景的png图像，这样如果圆和方块在小鸟后面也能显示出来。



太震撼了，我们已经开发了一个很酷的应用。不过正像前面说的，得靠你来实现电子商务等事宜。 →

www.aibbt.com 拼吾爱

there are no Dumb Questions

问：我们以前没有见过Image对象。你在画布上增加图像时用到了这个对象。这是什么？为什么我们不用document.createElement(“img”)来创建这个对象呢？

答： 问得好。你提到的方法也能创建图像对象。JavaScript Image构造函数无疑更直接，可以从JavaScript创建图像，还允许我们对加载过程有更多的控制（比如我们能轻松地使用一个处理程序，从而在图像加载时得到通知）。

所以说，在这里我们的目标是创建一个图像，另外在画布上绘制这个图像之前先要确保它已经加载。要达到这个目的，Image对象是最合适的。

问：画布很酷…… 不过与HTML相比也有些麻烦。除了基本形状之外，其他更复杂的图形绘制起来肯定很困难。

答： 毫无疑问，编写画布代码时实际上就是在编写图形代码。浏览器会帮你负责很多细节问题，比如让元素浮在页面上，使你不用操心一切都由自己来绘制。与浏览器不同，使用画布时，你必须告诉画布所有内容应该放在哪里。

但是，画布能赋予你超强的能力，可以绘制你能想象的几乎所有图形（目前还只是2D图形）。而且现在还只是画布的早期阶段，将来可能还会出现一些JavaScript代码库，利用这些代码库可以更容易地编写在画布上绘制图形的代码。

问：我注意到，对于很长的微博，超出画布边界的微博会消失，这个问题怎么解决？

答： 要修正这个问题，一种方法是查看微博中包含多少个字符，如果大于某个数，就把它分为多行，单独地把各行绘制到画布中。我们在wickedlysmart.com上提供的代码中已经包含了一个名为splitIntoLines的函数，可以利用这个函数来完成这个工作。

问：我还注意到，有些微博中包含HTML实体，比如"和&。这些是什么？

答： 用来得到微博JSON的Twitter API会把人们在微博中提交的字符转换为HTML实体。这是一件好事，因为有些特殊字符（或者甚至引号）可能会影响我们从JSON正确地得到微博，所有这些特殊字符都会表示为实体。如果我们用HTML显示微博，这些实体就会在浏览器中显示为你要看到的字符，就像你向页面中增加的实体会在浏览器中正确地显示一样。不过，可以看到，在画布中它们看起来可就不太妙了。遗憾的是，目前画布API中还没有函数能够把这些实体转换回原来的字符，所以你必须自己来完成。

问：能不能在画布中做些有意思的事情，比如对文本或形状加阴影？

答： 当然可以！利用画布你可以做很多有趣的事情，阴影当然是其中之一。可以想见，只需对上下文设置一些属性就能创建一个阴影。例如，要设置阴影的滤镜大小，可以设置context.shadowBlur。可以用context.shadowOffsetX和context.shadowOffsetY设置阴影的位置，另外用context.shadowColor可以设置阴影的颜色。

利用画布还可以做很多其他事情，你可能想试试，比如绘制梯度、旋转形状，在矩形上加上圆角等。

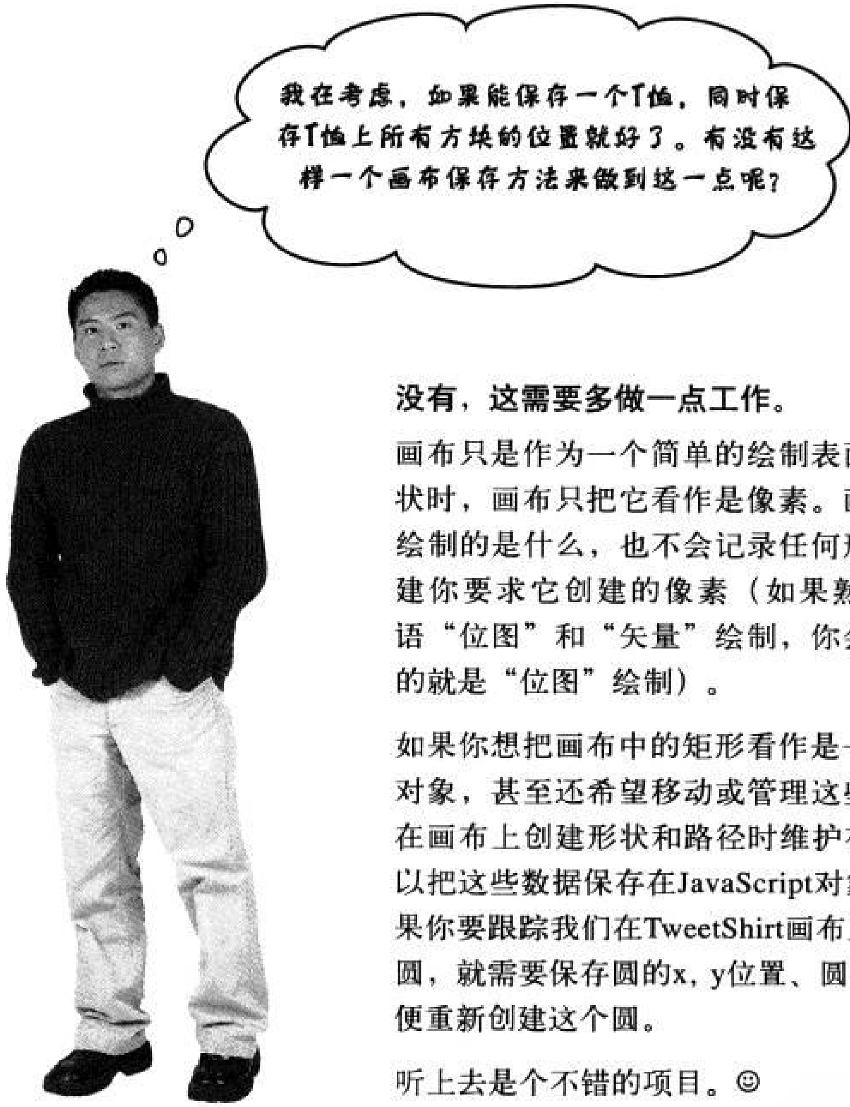
问：用画布还能做哪些有意思的事情？

答： 太多了！后面的几章我们会介绍另外几种使用画布的方法，如果要了解更多，很有必要看看画布AP：http://dev.w3.org/html5/2dcontext/。

问：这些画布代码在我的移动设备上也能正常工作吗？我是不是还得为移动用户重写代码？

答： 如果你的移动设备有一个现代浏览器（比如Android、iPhone和iPad等等都会提供这样一个浏览器），那么这些代码完全可以正常工作（页面的大小可能不太合适，不过功能没有问题）。画布的妙处就在于你所画的东西在任何地方（只要是支持画布的浏览器）看起来都是一样的（因为你在用原始像素绘制）。幸运的是，现代的智能设备（比如Android、iPhone和iPad）都提供了先进的浏览器，已经具备桌面浏览器的大部分功能。

pin51.com 拼吾爱 最新最全资源分享下载站



没有，这需要多做一点工作。

画布只是作为一个简单的绘制表面。绘制一个形状时，画布只把它看作是像素。画布并不关心你绘制的是什么，也不会记录任何形状。它只是创建你要求它创建的像素（如果熟悉图形图像术语“位图”和“矢量”绘制，你会发现画布所做的就是“位图”绘制）。

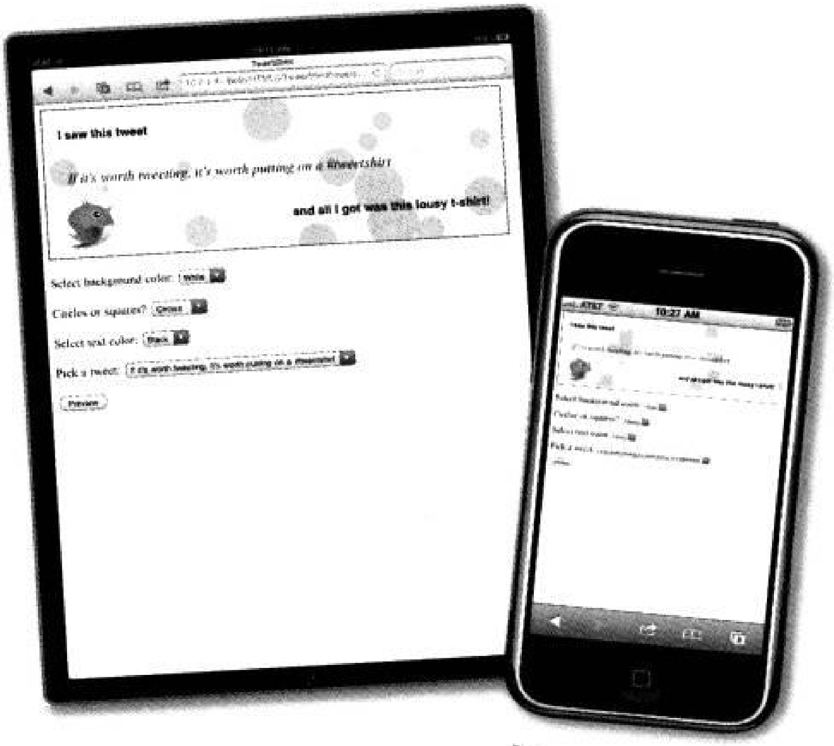
如果你想把画布中的矩形看作是一组可以保存的对象，甚至还希望移动或管理这些对象，就需要在画布上创建形状和路径时维护有关的信息。可以把这些数据保存在JavaScript对象中。例如，如果你要跟踪我们在TweetShirt画布上绘制的随机的圆，就需要保存圆的x，y位置、圆半径和颜色，以便重新创建这个圆。

听上去是个不错的项目。☺

拼吾爱

祝贺你们，你们成功了！它甚至在我的iPad上也能正常工作，所以这对于经常出行的顾客来说简直太完美了。我真兴奋。我们在开了一个TweetShirt启动派对，一起来吧。





TweetShirt创始人还想告诉你，她很高兴看到这个Web应用在她的iPad和iPhone上也能很好地工作！如果她满意，我们当然也高兴。

pin51.com 拼吾爱 最新最全 免费下载

BULLET POINTS

- 画布是一个元素，可以放在页面上来创建一个绘制空间。
- 除非你指定，否则画布没有默认的样式或内容（所以在画布上绘制内容或用CSS增加边框之前，在页面上是看不到画布的）。
- 页面上可以有多个画布。当然，需要为每个画布提供一个唯一的id，以便使用JavaScript分别访问。
- 要指定画布元素的大小，可以使用元素的width和height属性。
- 画布上的所有内容都使用JavaScript绘制。
- 要在画布上绘制，首先需要创建一个上下文。目前，2D上下文是唯一的选择，不过将来可能还会有其他上下文类型。
- 要在画布中绘制，需要有一个上下文，因为上下文提供了一个特定的接口（例如，2D或3D）。你可以从多种接口中选择来完成画布上的绘制。
- 使用上下文属性和方法来访问画布。
- 要在画布中绘制一个矩形，可以使用context.fillRect方法。这个方法会创建一个矩形，并填充指定的颜色。
- 要创建一个矩形轮廓，可以使用strokeRect而不是fillRect。
- 使用fillStyle和strokeStyle可以改变默认的填充和笔划颜色，默认颜色为黑色。
- 可以使用CSS中同样的格式指定颜色（例如，“black”、“#000000”、“rgb(0, 0, 0)”）。记住要在fillStyle值两边加上引号。
- 并没有一个fillCircle方法。要在画布上绘制一个圆，需要绘制一个弧。
- 要创建任意的形状或弧，首先需要创建一个路径。
- 路径是一个不可见的线或形状，它定义了画布上的一条线或区域。用笔划描出路径或填充路径之前，路径是看不到的。
- 要创建一个三角形，可以使用beginPath创建一个路径，然后用moveTo和lineTo来绘制路径。使用closePath可以连接路径上的两个点。
- 要绘制一个圆，可以创建一个360°的弧。起始角为0，终止角为360°。
- 画布中使用弧度来指定角，而不是使用度，所以需要从度转换为弧度来指定起始角和终止角。
- 360度 = 2Pi弧度。
- 要在画布上绘制文本，可以使用fillText方法。
- 在画布中绘制文本时，需要使用上下文属性指定位置、样式和其他属性。
- 设置一个上下文属性时，它会应用到后面的所有绘制操作，直到你再次改变这个属性。例如，改变fillStyle会影响设置fillStyle之后绘制的所有形状和文本的颜色。
- 可以用drawImage方法向画布增加图像。
- 要增加一个图像，首先需要创建一个图像对象，并确保它完全加载。
- 在画布上绘制就像在图形程序中完成“位图”绘制。

pin51.com 拼吾爱

Web镇问询报

独家新闻，<canvas>和<video>实际上已成一体！

Web镇——首家报道

经过深入采访，终于能告诉大家，<canvas>和<video>并不只是共享相同的Web页面，他们还有更多秘密……可以说，它们已经把内容混合在一起了。

Troy Armstrong报道
问询报特约撰稿人



当地居民Heidi Musgrove听到关于这两个元素的真相时非常震惊。

<Video>如是说，"没错，我们关系很铁。要知道，我是个很简单的人，我知道怎么显示视频，而且这方面很擅长。不过我能做的只有这些。有了<canvas>，一切都改变了。我有了定制控件，我能过滤我的视频内容，我还能立即显示多个视频。"

我们请<canvas>做出评论。她是不是站在<video>标记背后的女人？<Canvas>告诉我们，"嗯，<video>自己就很棒，你们很清楚，他要对所有这些视频编码译码器解码，保持他的每秒帧数，等等等等，这可是个艰巨的任务，我就做不来。不过，有了我，他终于可以摆脱原来（抱歉这么讲）"呆板"的视频回放方式。我为他提供了一种方法，可以采用各种有创意的方式将视频混合到Web体验中。"

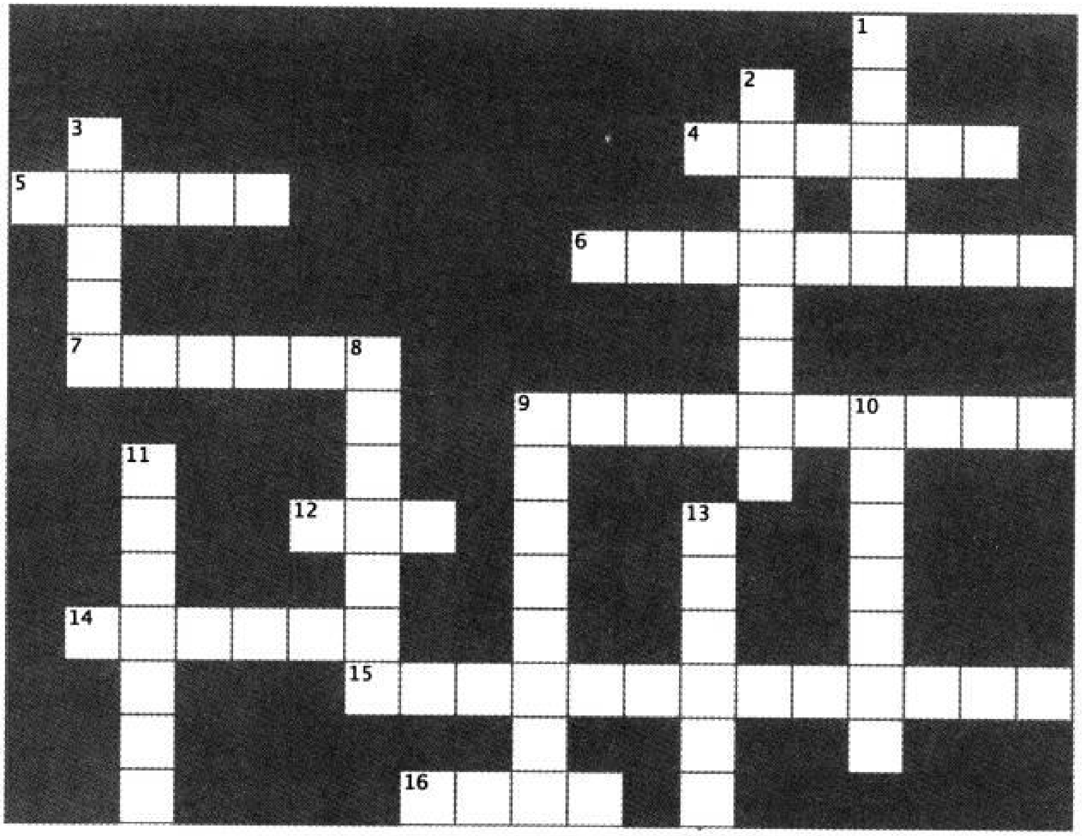
是啊，谁能想到呢？在我看来，利用这种新关系，我们会得到一些有趣的东西，这就是<canvas>+<video>！

这个事实批露后，会带来一系列的影响，这些内容肯定还会在视频一章中出现，那时他们的这种"情侣"关系就会暴露在公众视野中了。

pin51.com 拼吾爱 最新最全的下载站

HTML5填字游戏

我们迫切希望到下一章看看关于<canvas>和<video>的独家报道。同时，先做个小小的填字游戏，还可以来点茶，巩固你新学的画布知识。



横向

4. 画布上的一切都是 __________ 。
5. 我们将“and all I got was this lousy t-shirt!”文本 ________ 对齐。
6. 用一种颜色填充形状时要设置的属性。
7. 可以使用一个_________ 处理程序指出某个内容何时完成加载。
9. Jim尝试用来创建圆的不存在的上下文方法。
12. 要用一个______ 画圆。
14. 可以用这种方法让一个形状的路径可见。
15. 想知道选择了哪个选项吗？你可能需要这个属性。
16. 画一个形状时所创建的一个不可见的线。

纵向

1. 放微博最合适的地方。
2. 这个上下文方法会创建一个矩形。
3. 画布和 ________相处得很好。
8. 圆有360 ______ 。
9. 使用这个方法可以在画布上绘制文本。
10. 我们按度考虑，画布按__________ 考虑。
11. 这个对象包含一些方法和属性，可以用来在画布上完成绘制。
13. 将路径铅笔移动到点100, 100，可以使用 _____ (100, 100);

pin51.com 拼吾爱

扮演浏览器答案

既然已经有了界面，执行这些JavaScript语句，并写出各个界面元素的值。

假设你使用这个界面为你的T恤取值。



```
var selectObj = document.getElementById("backgroundColor");
var index = selectObj.selectedIndex;
var bgColor = selectObj[index].value;                    white

var selectObj = document.getElementById("shape");
var index = selectObj.selectedIndex;
var shape = selectObj[index].value;                      circles

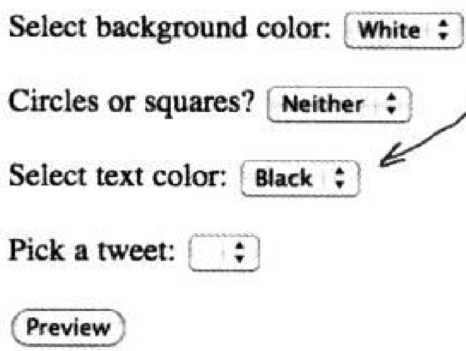
var selectObj = document.getElementById("foregroundColor");
var index = selectObj.selectedIndex;
var fgColor = selectObj[index].value;                    black
```

注意，对于每个菜单选项值，我们会得到包含这个选项的选择元素，用selectedIndex属性找到所选的选项，然后得到所选选项的值。

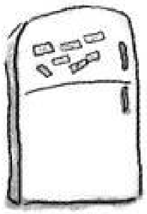
记住，选项的值可能与你在控件中看到的文本不同。在这里，文本的首字母大小就不同。

如果需要，可以再看看HTML，查看这些选项的值。

这些是TweetShirt界面中创建以上答案所选的值。



pin51.com 拼吾爱



伪代码磁贴答案

利用你的伪代码能力组织下面的伪代码。我们需要为drawSquare函数编写伪代码。这个函数取一个画布和一个上下文参数，会在画布上绘制一个大小随机的方块。以下是我们的答案。

我们已经帮你填入了这个磁贴。

```
function drawSqua[canvas], [context] ){
    [为方块计算一个随机的宽度]
    [为画布中的方块计算一个随机的x位置]
    [为画布中的方块计算一个随机的y位置]
    [将fillStyle设置为“lightblue”]
    [在位置x,y画一个宽度为w的方块]
}
```

你的磁贴放在这里！

Sharpen your pencil Solution

为了确保每次点击预览按钮时在画布上只看到新的方块，需要使用用户在“backgroundColor”选择菜单中选择的背景色来填充画布的背景。首先，实现一个函数用这个颜色填充画布。请在下面填空，完成这个代码。以下是我们的答案。

```
function fillBackgroundColor(canvas, context) {
    var selectObj = document.getElementById("backgroundColor");
    var index = selectObj.selectedIndex;
    var bgColor = selectObj.options[index].value;
    context.fillStyle = bgColor;
    context.fillRect(0, 0, canvas.width, canvas.height);
}
```

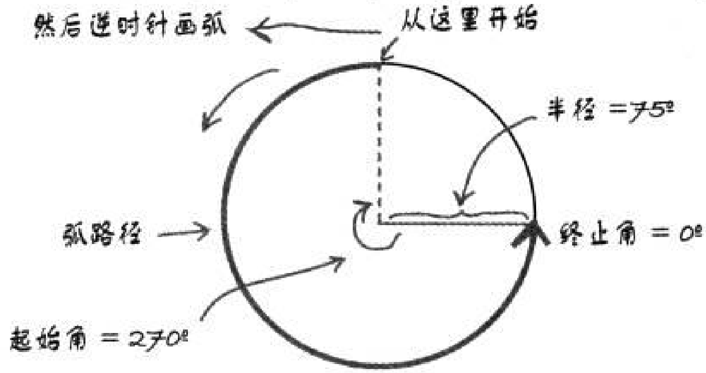
要创建一个背景色，我们要做的就是绘制一个矩形，用一种颜色填充整个画布。

扮演浏览器答案

解释这个arc方法调用，在圆上画出所有值，包括这个方法创建的路径。



```
context.arc(100, 100, 75, degreesToRadians(270), 0, true);
```



Exercise Solution

现在已经有了路径！然后呢？

当然要使用这个路径来画线，并用颜色填充你的形状！下面创建一个简单的HTML5页面，其中包含一个画布元素，键入前面的所有代码。再运行试试看。

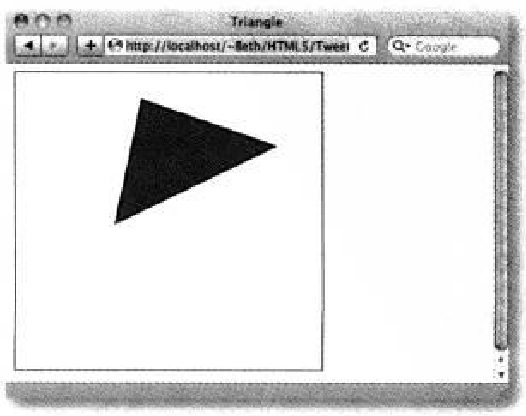
```
context.beginPath();
context.moveTo(100, 150);
context.lineTo(250, 75);
context.lineTo(125, 30);
context.closePath();
```

这是目前为止的代码。

```
context.lineWidth = 5;
context.stroke();
context.fillStyle = "red";
context.fill();
```

代码	说明
context.lineWidth = 5;	设置线宽用来在路径上画线。
context.stroke();	用线描路径。
context.fillStyle = "red";	设置颜色来用红色填充三角形。
context.fill();	用红色填充三角形。

加载三角形页面时，我们会得到这个结果（我们创建了一个可以用来绘制的300 x 300的画布）。

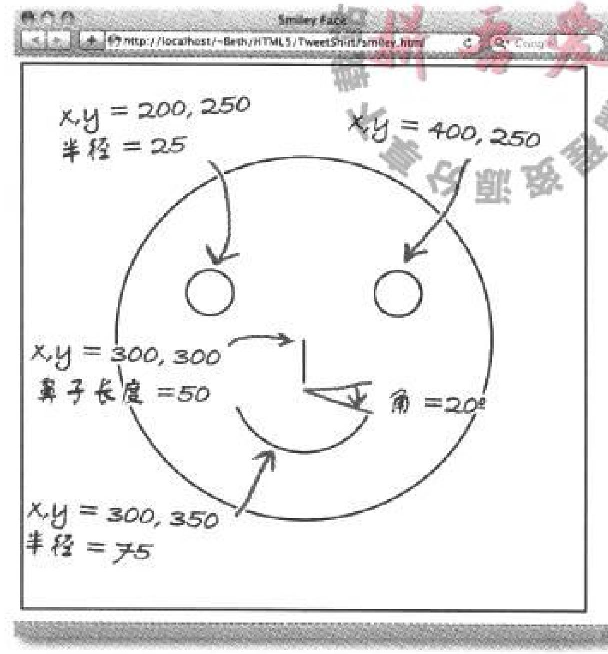


www.aibbt.com 让未来触手可及



中场休息答案

现在发挥你的画弧和路径的技能来创建一个笑脸。用你需要的代码在下面填空，完成这个笑脸。我们已经给出了一些提示，告诉你应该在图中哪里画眼睛、鼻子和嘴。



下面是我们的答案：

```
function drawSmileyFace() {
    var canvas = document.getElementById("smiley");
    var context = canvas.getContext("2d");

    context.beginPath();
    context.arc(300, 300, 200, 0, degreesToRadians(360), true);
    context.fillStyle = "#ffffcc";
    context.fill();
    context.stroke();
```

圆脸。这部分我们帮你完成了。注意这个圆用黄色填充。

```
    context.beginPath();
    context.arc(200, 250, 25, 0, degreesToRadians(360), true);
    context.stroke();
```

这是左眼。圆心在x=200，y=250，半径为25，起始角为0，终止角是Math.PI × 2弧度（360度）。我们用笔划描这个路径，来得到圆的轮廓（而不是填充）。

```
    context.beginPath();
    context.arc(400, 250, 25, 0, degreesToRadians(360), true);
    context.stroke();
```

这是右眼。与左眼类似，只不过位置在x=400。这里方向使用了逆时针（true）（不过对于一个完整的圆来说，方向并不重要）。

```
    context.beginPath();
    context.moveTo(300, 300);
    context.lineTo(300, 350);
    context.stroke();
```

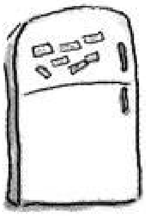
这是鼻子。我们使用moveTo(300,300)将笔移动到x=300，y=300开始画线。然后使用lineTo(300,350)，因为鼻子长度为50。然后用笔划描这个路径。

为了得到一个更真实的微笑，我们让嘴的起始角和终止角分别为x轴以下20度。这说明起始角为20°，终止角为160°。

```
    context.beginPath();
    context.arc(300, 350, 75, degreesToRadians(20), degreesToRadians(160), false);
    context.stroke();
}
```

方向是顺时针（false），因为我们希望这是一个微笑着的嘴。（记住，起点在嘴中心的右边）。

拼吾爱



代码磁贴答案

这是你第一次尝试画布文本。下面我们给出了drawText的部分代码，我们将调用这个方法在预览画布上绘制所有文本。看看你能不能完成这个代码，在画布上绘制“I saw this tweet”和“and all I got was this lousy t-shirt”，绘制具体微博的代码以后再完成。以下是我们的答案。

```
function drawText(canvas, context) {

    var selectObj = document.getElementById("foregroundColor");

    var index = selectObj.selectedIndex;

    var fgColor = selectObj[index].value;

    context.fillStyle = fgColor;

    context.font = "bold 1em sans-serif";

    context.textAlign = "left";

    context.fillText("I saw this tweet", 20, 40);


    // Get the selected tweet from the tweets menu

    // Draw the tweet


    context.font = "bold 1em sans-serif";

    context.textAlign = "right";

    context.fillText("and all I got was this lousy t-shirt!",
        canvas.width-20, canvas.height-40);

}
```

提示：这是“I saw this tweet”文本的x,y位置。

目前，我们只是在这里给出注释，将来会在这里放入绘制微博文本的具体代码。

提示：微博将用一种斜体serif字体，不过我们希望这个文本使用粗体sans-serif字体。

提示：我们希望这个文本放在右下角。

我们希望在距画布右边20像素、距底边40像素的位置绘制这个文本，以便与第一行文本平衡。

多余的磁贴。

fillCircle　fillRect　left

pin51.com 拼吾爱

Exercise Solution

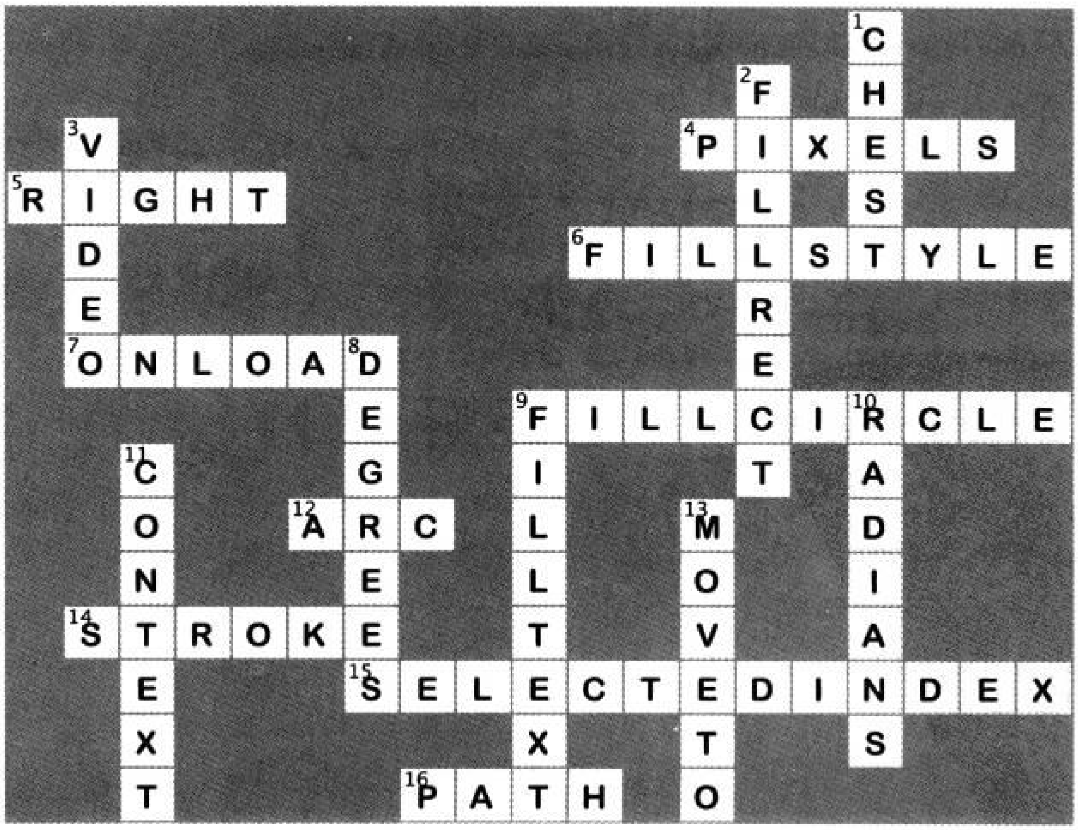
看看你能不能利用Judy给出的代码完成drawBird函数。drawBird函数取一个画布和上下文参数，在画布上绘制一个小鸟。假设利用这个函数我们将把“twitterBird.png”放在x=20, y=120位置上，宽度和高度都为70。我们已经帮你写出了这个方法的声明和第一行代码。下面是我们的答案。

别忘了要在previewHandler函数中增加一个drawBird函数调用。

你的代码写在这里。

```
function drawBird(canvas, context) {
    var twitterBird = new Image();
    twitterBird.src = "twitterBird.png";
    twitterBird.onload =function() {
     context.drawImage(twitterBird, 20, 120, 70, 70);
    };
}
```

HTML5填字游戏答案



mn51.com 拼吾爱 最新最全资源下载站



TweetShirt复活节彩蛋

你已经做出了完美的TweetShirt预览—现在做什么呢？嗯，如果你真的想完成自己的T恤设计，那么肯定能做到！怎么做呢？下面再额外布置一个小小的附加题，你可以对代码做些补充，如果愿意，可以增加一个TweetShirt"复活节彩蛋"。根据你的设计创建一个图像，把它上传到一个能够在T恤上打印图像的网站（Web上有很多这样的网站）。

怎么做到这一点呢？很简单！我们可以使用canvas对象的toDataURL方法。如下：

我们建立了一个新函数makeImage来增加这个功能。

```
function makeImage() {
    var canvas = document.getElementById("tshirtCanvas");
    canvas.onclick =function () {
        window.location = canvas.toDataURL("image/png");
    };
}
```

得到画布对象……

增加一个事件处理程序，当点击画布时，它会创建这个图像。

将浏览器窗口位置设置为所生成的这个图像，这样就会看到浏览器页面中只有这个图像。

我们要求画布创建一个png图像（在画布上绘制像素）。

注意，png是浏览器唯一肯定支持的格式，所以建议你使用这种格式。

现在，只需要在窗口onload函数中增加一个makeImage调用，这样一来，当你点击画布时就会创建一个图像了。可以试一试。如果你做出了自己的T恤一定要让我们知道。

```
window.onload =function() {
    var button = document.getElementById("previewButton");
    button.onclick = previewHandler;
    makeImage();
}
```

调用makeImage为画布增加点击事件处理程序。这就完成了你的复活节彩蛋。



Watch it!

如果从file://运行代码，有些浏览器不允许你从画布获取图像。

如果希望这个代码在所有浏览器上都通用，要从localhost:// 或托管服务器运行这个代码。

pin5i.com 拼吾爱 最新编程资源下载站

pin51.com 拼吾爱

8 不再是父辈的老电视

视频

……以及特邀演员"画布"



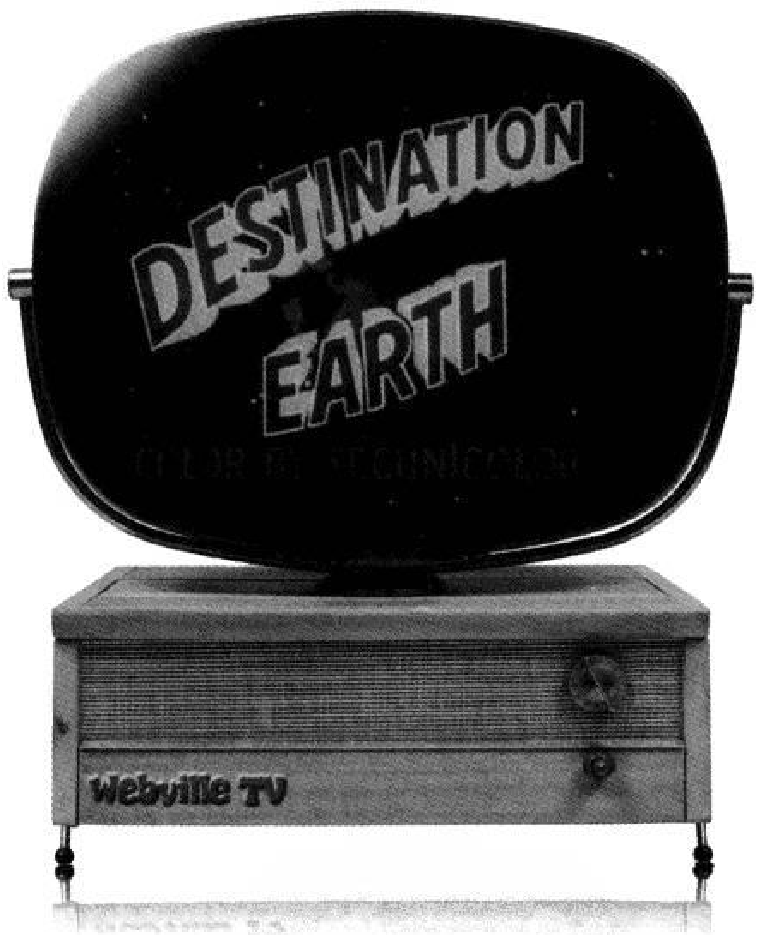
不再需要插件。毕竟，视频现在是HTML家族的直系成员。把<video>元素直接放在页面中，你就能立即看到视频，而且大多数设备上都能支持。不过，视频绝不仅仅是一个元素，它还是一个JavaScript API，有了它，我们可以控制视频的播放、创建自己的定制视频界面，还可以用全新的方式集成视频与HTML的其余部分。说到集成……还记得我们一直在谈论的视频与画布的联系吧—你会看到，视频和画布在一起会为我们提供一种强有力的新方法来实时地处理视频。这一章我们先在页面中玩转视频，然后再来研究JavaScript API。来吧，只用一点点标记、JavaScript、视频与画布，结果会让你瞠目结舌。

pin51.com 拼吾爱 最新最全资源分享下载站

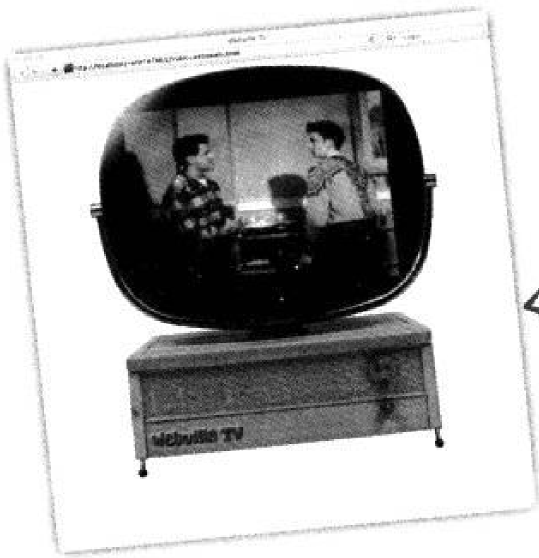
认识Webville TV

Webville TV，你想要的所有内容尽在其中，比如《Destination Earth》、《The Attack of the 50′ Woman》、《The Thing》、《The Blob》，不难理解还会有一些50年代的教育片。Web镇上你还能期待些什么呢？不过，这只是内容方面，在技术方面难道你不期待HTML5视频吗？

当然，这只是愿景，如果你希望愿望成真，就必须动手来构建Webville TV。在接下来的几页，我们会从头开始，利用HTML5标记和video元素，再加一些JavaScript来构建Webville TV。



用100%的HTML5技术构建Webville TV。



要有一个理解你的浏览器！

先搞定HTML部分……

嘿，这已经是第8章了，所以我们别再浪费时间！直接创建一些HTML吧：

```
<!doctype html>
<html lang="en">
<head>
  <title>Webville TV</title>
  <meta charset="utf-8">
  <link rel="stylesheet" href="webvilletv.css">
</head>
<body>
<div id="tv">
   <div id="tvConsole">
      <div id="highlight">
          <img src="images/highlight.png" alt="highlight for tv">
      </div>
      <div id="videoDiv">
          <video controls autoplay src="video/preroll.mp4" width="480" height="360"
                poster="images/prerollposter.jpg" id="video">
          </video>
      </div>
   </div>
</div>
</body>
</html>
```

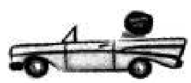
很标准的HTML5。

别忘了CSS文件，让它看上去漂亮一些。

只需一个小图像，就能让它看起来像是一个真正的电视机。

这是用来播放视频的<video>元素。稍后会更详细地介绍……

插上电视，试一试……



这里需要明确几件事情：首先，要确保上面的代码都已经键入到一个名为webvilletv.html的文件中；其次，要确保已经下载了CSS文件，最后，还要确保下载了视频文件，并把它们放在一个名为video的目录中。如果以上都做到了，加载这个页面，坐下来好好欣赏吧。

所有内容可以从http://wickedlysmart.com/hfhtml5下载。

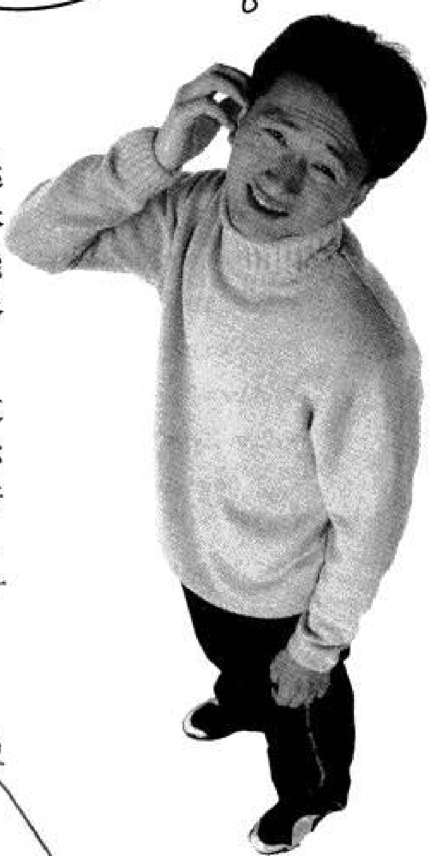
这是我们看到的。注意，如果把鼠标停在屏幕上，会出现一组控件，可以用这些控件来暂停和播放视频，还可以设置音频或者在视频中搜索。

如果你遇到问题，请翻开下一页！



pin51.com 拼吾爱

我没看到任何视频啊。我反复检查了代码，而且视频文件也放在正确的目录下了。怎么回事呢？



嗯，可能是视频格式的缘故。

尽管浏览器制造商对HTML5中的<video>元素和API达成了一致，但是并非所有人对于视频文件本身的具体格式都意见相合。例如，如果你使用的是Safari，可能倾向于H.264格式。如果你的浏览器是Chrome，首选格式则是WebM，诸如此类。

等你读到这些文字时，这些格式在所有浏览器上可能会得到更广泛的支持。所以，如果你的视频能正常工作，那太好了。一定要时刻关注Web，这个主题还在不断演进，需要了解它的最新进展。稍后我们还会回来进一步讨论这个主题。

在我们刚才写的代码中，我们假设格式是H.264，这在Safari、Mobile Safari和IE9+中是可以的。如果你在使用其他浏览器，可以查看你的video目录，会看到不同类型的视频，分别有3种不同的文件扩展名：“.mp4”、“.ogv”和“.webm”（稍后就会介绍这些扩展名的含义）。

对于Safari，应该使用 .mp4（包含H.264）。

对于Google Chrome，则应使用.webm格式，要把src属性替换为：

```
src="video/preroll.webm"
```

如果你在使用Firefox或Opera，就要把src属性替换为：

```
src="video/preroll.ogv"
```

你可以试一试，我们稍后还会再做说明。

如果你使用的是IE8或更早版本，那你太不走运了。要知道，这已经是第8章了！你怎么还能使用IE8或更早版本呢？赶快升级！不过，如果想知道如何为你的IE8用户提供备用内容，别着急，我们会谈到的。

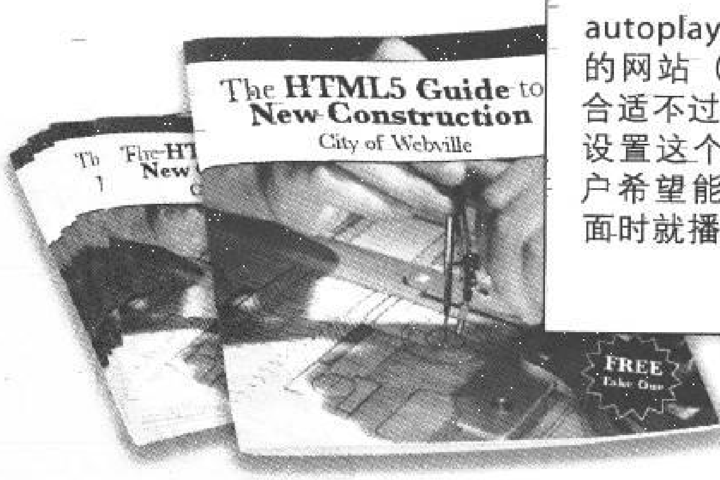
video元素如何工作？

现在你已经在页面上播放了一个视频，但是在继续学习后面的内容之前，先退一步，看看标记中使用的这个video元素。

```
<video controls
       autoplay
       src="video/preroll.mp4"
       width="480"  height="360"
       poster="images/prerollposter.jpg"
       id="video">
</video>
```

- controls：如果有这个controls属性，播放器会提供控件，允许用户控制视频和音频的播放。
- autoplay：如果有autoplay属性，页面一旦加载视频就开始播放。
- src：视频的源位置。
- width、height：页面中视频的宽度和高度。
- poster：不播放电影时显示的海报图片。
- id：video元素的id，以后可以利用这个id从JavaScript访问这个元素。

Web镇HTML5指南提供的另一个有用的技巧。



视频礼仪：autoplay属性

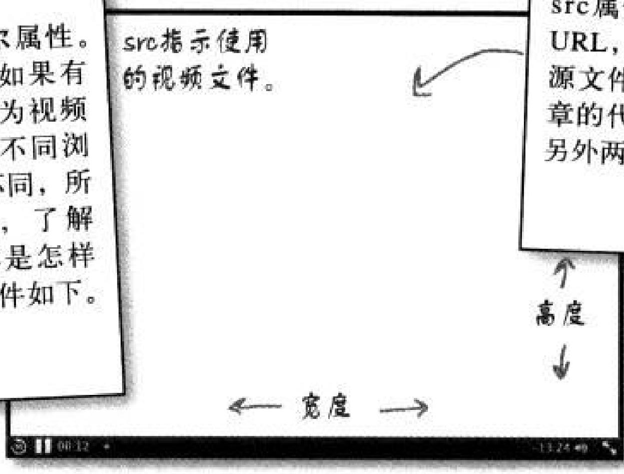
autoplay对于YouTube和Vimeo之类的网站（或者WebvilleTV）可能再合适不过了，不过在<video>标记中设置这个属性前请三思。通常，用户希望能够参与决定是否在加载页面时就播放视频。

深入研究视频属性……

下面更深入地分析video元素中一些比较重要的属性：

controls

controls属性是一个布尔属性。可以有，也可以没有。如果有这个属性，浏览器就会为视频显示区增加内置控件。不同浏览器提供的控件会有所不同，所以请查看各个浏览器，了解不同浏览器的控件具体是怎样的。Safari浏览器中的控件如下。



src

src属性类似于<img>元素的src属性—这是一个URL，告诉video元素在哪里查找源文件。在这里，源文件为video/preroll.mp4。（如果已经下载这一章的代码，你会在video目录中找到这个视频以及另外两个视频）。

preload

布尔属性preload通常用来细粒度地控制如何加载视频来实现优化。大多数情况下，浏览器会根据一些因素来选择加载多少视频，比如是否设置了autoplay，另外用户的带宽是多少。可以覆盖这个设置，把preload设置为none（在用户选择“播放”之前不下载任何视频），或者设置为metadata（只下载视频元数据，而不下载视频内容），还可以设置为auto，让浏览器来做决定。

autoplay

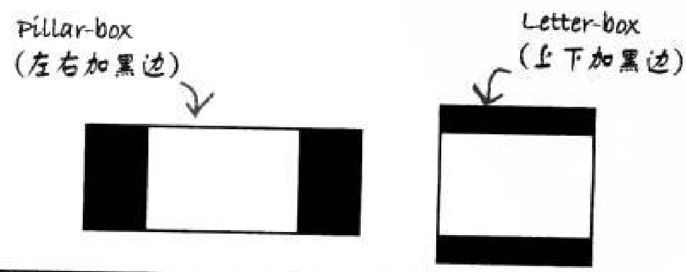
autoplay布尔属性告诉浏览器：一旦有了足够的数据就开始播放视频。你可能会看到，我们的演示视频几乎立即开始播放。

poster

浏览器通常会显示一个视频帧作为“海报”图片来表示这个视频。如果删除了autoplay属性，点击播放之前就会看到显示这个图片。要由浏览器来选择显示哪一帧；通常浏览器都会显示视频的第一帧…… 这往往是黑屏。如果想显示一个特定的图片，就要由你来创建一个要显示的图片，并使用poster属性指定这个图片。

width, height

width和height属性设置视频显示区（也称为“视窗”）的宽度和高度。如果指定了一个poster属性，海报图片就会缩放为你指定的这个宽度和高度。视频也会缩放，不过仍保留原来的宽高比（例如，4:3或16:9），所以，如果两边或者上下有多余的空间，视频会采用上下加黑边的模式（Letter-boxed)或左右加黑边的模式（Pillar-boxed）来适应显示区大小。如果想得到最佳性能，就应该采用视频的原始大小（这样浏览器就不必实时地缩放视频了）。



loop

这也是一个布尔属性，loop属性会使视频播放结束后自动地重新开始。

pin51.com 拼吾爱 最新最全资源分享下载站



是的，每个浏览器中HTML视频的控件确实不同。

控件的外观由实现浏览器的人决定。在不同的浏览器和不同的操作系统上，这些控件看上去确实各不相同。有些情况下，例如在小平板设备上，它们的外观和表现就会不同，因为这些设备的工作方式就不同（好在这个问题已经替你考虑到了）。这个道理我们明白，比如说在各种桌面浏览器上，能有一致的控件是一件好事，但是这并不算HTML5规范中正式的部分，而且有些情况下，可能某个方法对一种操作系统适用，在面对另一个操作系统的用户界面原则时却会完全失效。所以，要知道控件可能会不同，另外如果你确实觉得有必要，也可以为你的应用实现定制控件。

后面就会这样做……

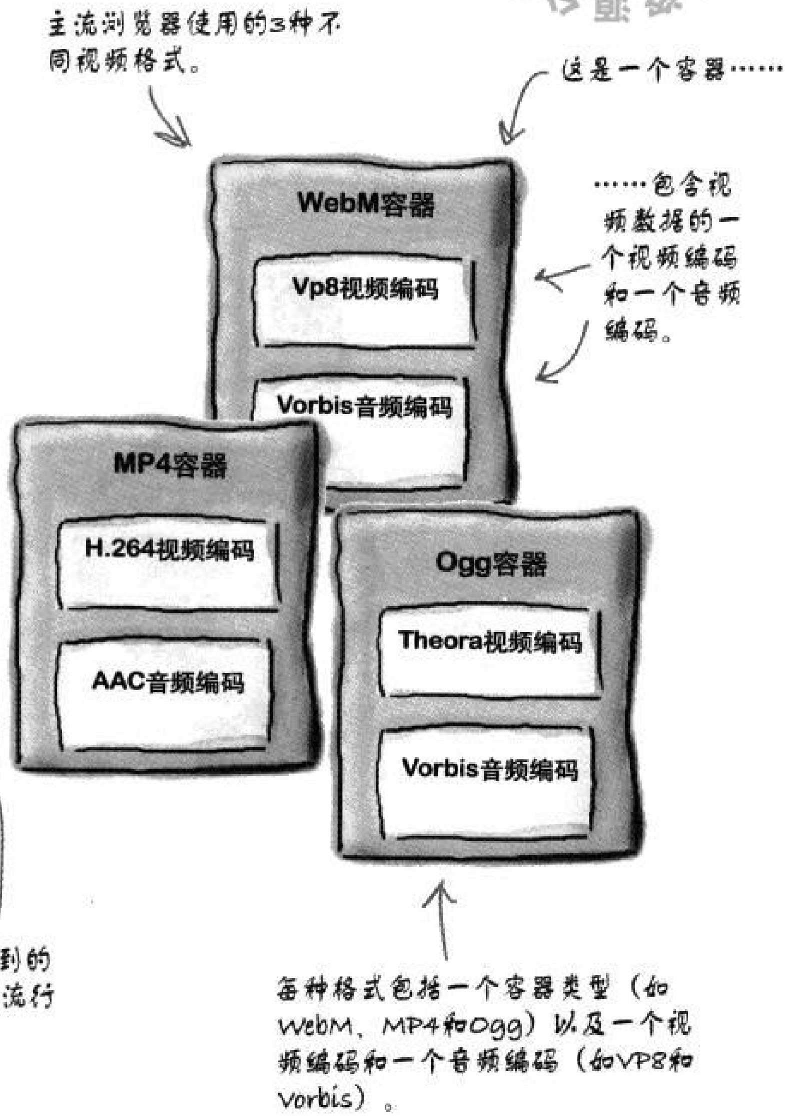
pin51.com 拼吾爱

关于视频格式需要知道什么

我们真希望一切都像video元素及其属性那样干净整洁，可惜事实上Web上的视频格式相当混乱。视频格式是什么？可以这样来考虑：一个视频文件包含两部分，一个视频部分和一个音频部分，每个部分都使用一种特定的编码类型来编码（这样可以缩小数据大小，并能更高效地播放）。大多数情况下，没有一种得到大家共识的编码，有些浏览器制造商推崇H.264编码，另外一些却喜欢VP8，还有一些倾向于开源编码Theora。而且，包含视频和音频编码的文件（称为容器）也有自己的格式和格式名，这让情况更为复杂。所以我们面对的真像是一个混杂口味的“术语汤”。

不管怎样，如果所有浏览器制造商都能达成一致，在Web上使用同一种格式，这当然是一个让人欢欣鼓舞的大同世界，不过，嗯，出于技术上、政治上甚至哲学方面的一些原因，这是不现实的。但这里并不打算就此展开争论，我们只是希望你对这个主题有足够的了解，对于如何支持你的观众能够做出自己的决定。

下面来看目前流行的一些编码，现在主要有3个竞争对手在努力征服这个（Web）世界……



等你读到这本书时，你掌握到的情况可能会有所不同，因为流行的编码往往会随时间变化。

HTML5规范允许采用任何视频格式。具体支持哪些格式由浏览器实现来确定。

www.pin51.com 拼吾爱 最新资源分享下载站

竞争选手

事实上，如果你打算为范围更广的用户提供内容，就不能只提供一种格式。另一方面。如果你只关心（比如说）Apple iPad，那么只支持一种格式也是可以的。如今我们有3个主要的竞争对手，我们来一一认识一下：

MP4容器，包含H.264视频和AAC音频

H.264由MPEG-LA公司授权。

目前有多种H.264，每一种分别称为一个“profile”（等级或类）。

MP4/H.264得到了Safari和IE9+的支持。一些版本的Chrome也提供了支持。

WebM容器，包含VP8视频和Vorbis音频

WebM由Google设计用来处理VP8编码视频。

WebM/VP8得到了Firefox、Chrome和Opera的支持。

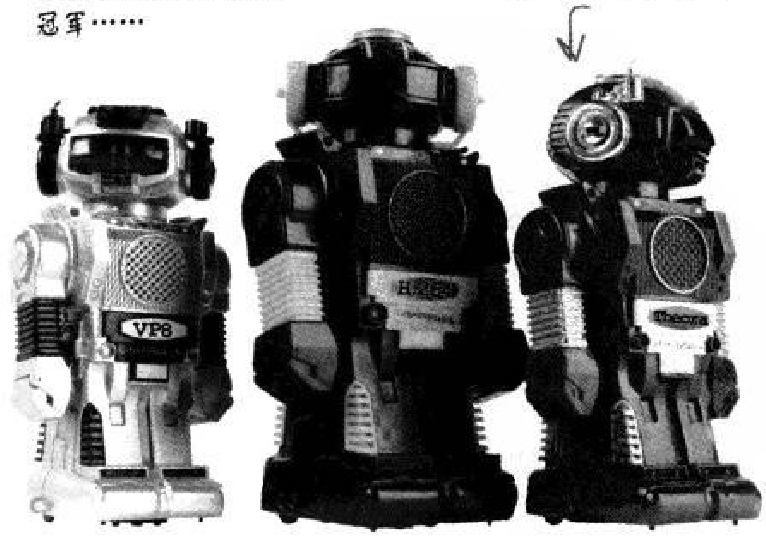
WebM格式的视频扩展名为.webm。

Ogg容器，包含Theora视频和Vorbis音频

Theora是一个开源编解码器。

采用Theora编码的视频通常包含在Ogg文件中，文件扩展名为.ogv。

Ogg/Theora得到了Firefox、Chrome和Opera的支持。



如何处理所有这些格式……

现在我们知道了，谈到视频格式，这真是一个混乱的世界，不过该怎么办呢？取决于你的观众，你可能决定只提供某一种视频格式，也可能提供多种。不论哪一种情况，可以在<video>元素中对应每种格式使用一个<source>元素（不要与src属性混淆），这样就能提供一组视频，每个视频分别有自己的格式，可以让浏览器选择它支持的第一种格式。如下：

注意我们从<video>标记删除了src属性……

……并增加了3个source标记，每个标记有自己的src属性，分别包含采用不同格式的视频版本。

```
<video src="video/preroll.mp4" id="video"
       poster="video/prerollposter.jpg" controls
       width="480" height="360">
    <source src="video/preroll.mp4">
    <source src="video/preroll.webm">
    <source src="video/preroll.ogv">
    <p>Sorry, your browser doesn't support the video element</p>
</video>
```

如果浏览器不支持视频，就会显示这个文本。

浏览器从上向下查找，直到找到它能播放的一种格式。

对于每个source元素，浏览器会加载视频文件的元数据来查看能不能播放这个视频（这个过程可能很耗费时间，不过我们可以在浏览器上更轻松地做到……请看下一页）。



BULLET POINTS

- 容器是用来包装视频、音频和元数据信息的文件格式。常用的容器格式包括：MP4、WebM、Ogg和Flash Video。
- 编解码器是用来对一种特定视频或音频编码进行编码和解码的软件。流行的web编解码器包括：H.264、VP8、Theora、AAC和Vorbis。
- 要由浏览器决定可以解码哪种格式的视频，不过并不是所有浏览器制造商都达成了一致，所以如果你想支持所有视频，就需要多种编码。

mn51.com 拼吾爱 最新最全下载站 分享资源

如何更具体地指定视频格式

可以告诉浏览器视频源文件的位置，使它能在不同版本中做出选择。不过，浏览器真正确定一个文件是否可以播放之前，还需要做一些检测工作。你可以为浏览器提供更多帮助，给出有关视频文件的MIME类型和编解码器（可选）的更多信息。

src中使用的文件实际上是具体视频（和音频以及一些元数据）的容器。

codecs参数指定使用哪个编解码器对视频和音频编码，来创建编码的视频文件。

视频编解码器。

音频编解码器。

```
<source src="video/preroll.ogv" type='video/ogg; codecs="theora, vorbis"'>
```

type是一个可选属性，这是向浏览器提供的一个提示，帮助它确定能不能播放这种类型的文件。

这是视频文件的MIME类型。指定了容器格式。

注意codecs参数的双引号。这说明type属性两边需要使用单引号。

可以更新<source>元素，让它包含我们的3种视频的类型信息，如下：

```
<video id="video" poster="video/prerollposter.jpg" controls width="480" height="360">
    <source src="video/preroll.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
    <source src="video/preroll.webm" type='video/webm; codecs="vp8, vorbis"'>
    <source src="video/preroll.ogv" type='video/ogg; codecs="theora, vorbis"'>
    <p>Sorry, your browser doesn't support the video element</p>
</video>
```

如果不知道codecs参数，可以省略，只使用MIME类型。这样效率会低一点，不过大多数情况下都能接受。

mp4的编解码器比另外两种更为复杂，因为h.264支持多种"等级"，对应不同使用情况（如高带宽和低带宽）会有不同的编码。所以，要想正确使用，需要了解如何对视频编码的更多细节。

如果你想完成自己的视频编码，就要对source元素中type参数使用的各种选择有更多了解。可以在http://wiki.whatwg.org/wiki/Video_type_parameters得到有关type参数的更多信息。

pin51.com 拼吾爱 最新最全资源分享下载站

there are no Dumb Questions

问：以后几年有没有希望协定一种容器格式或者编解码器类型？我们之所以建立标准不就是为了这个目的吗？

答： 可能不会很快有这样一种能够全盘统治的编码。正像我们前面所说的，这个主题夹杂着很多问题，各家公司都想在视频领域掌控自己的命运，另外还涉及很多复杂的知识产权方面的问题。HTML5标准委员会意识到了这一点，决定不在HTML5规范中指定视频格式。所以尽管从理论上讲HTML5支持（或者至少认识）所有格式，但实际上要由浏览器制造商来决定支持什么和不支持什么。

如果视频对你来说很重要，一定要关注这个主题。在接下来的几年里，能够看着情况逐渐明朗，这肯定是一件有趣的事情。而且，与往常一样，一定要记住你的用户需要些什么，另外一定要竭尽所能地提供支持。

问：如果我想对我自己的视频编码，从哪里开始呢？

答： 目前有很多视频采集和编码程序，选择哪一个取决于你在采集哪种视频，另外还要看你希望如何使用最终结果。有很多书整本介绍视频编码，所以你要做好准备，接下来会进入一个满是新的编略语和术语的世界。可以先从简单的开始，使用类似iMovie或Adobe Premiere Elements等程序，这些程序提供了Web视频编码的功能。如果你想用Final Cut Pro或Adobe Premiere做一些真正的视频处理工作，这些软件程序提供了它们自己的生产工具。最后，如果你从一个内容分发网络（Content Delivery Network，CDN）分发你的视频，会有很多CDN公司提供编码服务。所以你可以根据你的需要在众多方案中做出选择。

问：我能全屏播放我的视频吗？我很奇怪API中居然没有这样的一个属性。

答： 这个功能还没有得到标准化，不过如果在Web上搜索，你能找到很多方法可以在一些浏览器上做到这一点。有些浏览器提供了一个全屏控件（例如，平板设备），这就为video元素提供了这个功能。另外要记住，如果想办法实现了全屏，除了基本的播放外，你对视频所能做的处理可能会出于安全原因受到限制（当前的插件视频方案也存在同样的问题）。

问：视频的音量呢？我能用API来控制音量级别吗？

答： 当然能。可以把volume属性设置为一个介于0.0（静音）~1.0（最强音）之间的一个浮点值。任何时间都可以用你的视频对象设置这个属性：

```
video.volume = 0.9;
```

mln51.com 拼吾爱 最新最全电子书下载站

CASE FILE: VIDEO

TOP SECRET

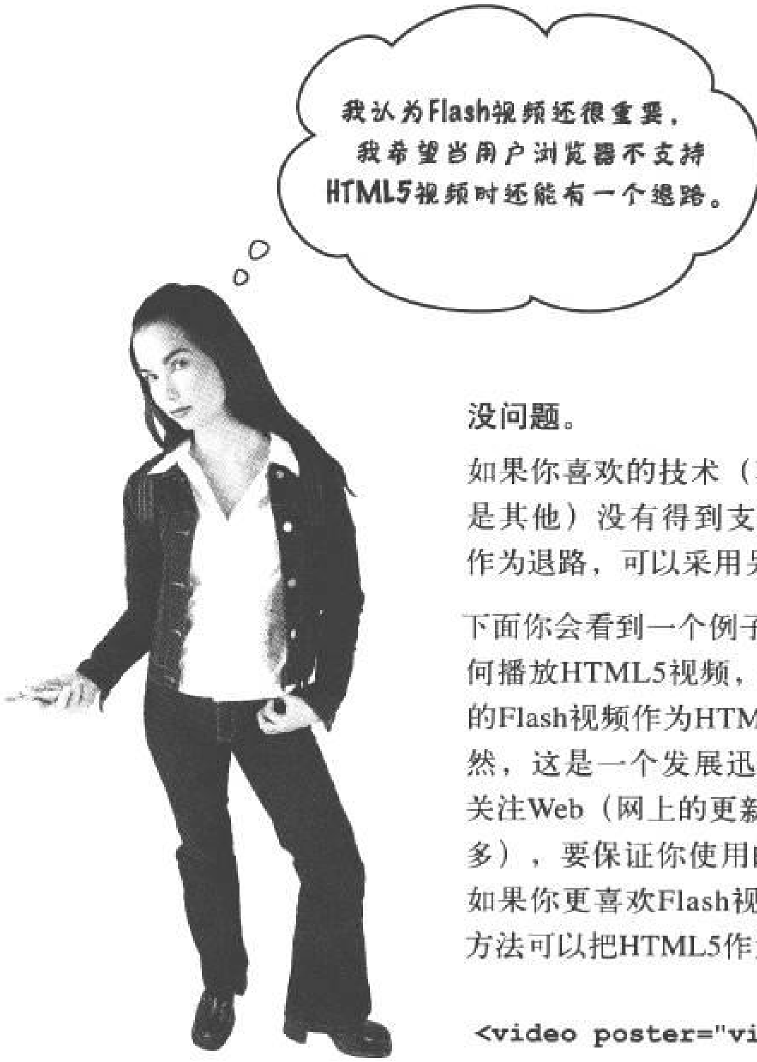
下一项任务：视频侦察

请四处走访一下，确定[redacted]各个浏览器对视频的支持水平（提示，这里给出几个网站，可以从中找到有关的最新信息：HTTP://EN.WIKIPEDIA.ORG/WIKI/HTML5_VIDEO，[redacted]HTTP://CANIUSE.COM/#SEARCH=VIDEO）。这里假设都使用浏览器的最新版本。对于每组浏览器/特性，如果得到支持就划勾，等你完成任务后，要给出报告作为你的下一个任务！

iOS和Android设备（以及其他）。 ↓ (Mobile Webkit)

视频 \ 浏览器	Safari	Chrome	Firefox	Mobile Webkit	Opera	IE9+	IE8	IE7 or <
H.264								
WebM								
Ogg Theora								

pin51.com 拼吾爱 最新最全资源分享下载站



没问题。

如果你喜欢的技术（不论是HTML、Flash或是其他）没有得到支持，还有一些技术可以作为退路，可以采用另一种视频播放器。

下面你会看到一个例子，假设浏览器不知道如何播放HTML5视频，这里将展示如何插入你的Flash视频作为HTML5视频的一个退路。显然，这是一个发展迅猛的领域，所以一定要关注Web（网上的更新要比书上的更新频繁得多），要保证你使用的是最新、最棒的技术。如果你更喜欢Flash视频，可能还会发现很多方法可以把HTML5作为后路，而不是Flash。

```
<video poster="video.jpg" controls>
        <source src="video.mp4">
        <source src="video.webm">
        <source src="video.ogv">
        <object>...</object>
</video>
```

在<video>元素中插入<object>元素。如果浏览器不认识<video>元素，就会使用<object>。

我听说有API

可以看到，用标记和<video>元素可以做很多事情。不过，<video>元素还提供了一个丰富的API，可以用来实现各种有趣的视频行为和体验。下面对你可能感兴趣的<video>元素方法、属性和事件做一个简短的总结（详细列表请参见规范）：



调用这些方法

- play ← 播放视频
- pause ← 暂停视频
- load ← 加载视频
- canPlayType ← 通过编程帮助你确定可以播放哪些视频类型。



使用这些属性

videoWidth	loop
videoHeight	muted
currentTime	paused
duration	readyState
ended	seeking
error	volume

这些都是<video>元素对象的属性。有些可以设置（如loop和muted），有些是只读的（如currentTime和error）。



捕获这些事件

play	abort
pause	waiting
progress	loadeddata
error	loadedmetadata
timeupdate	volumechange
ended	

这些是可以处理的事件，如果增加事件处理程序，出现你监听的事件时就会调用相应的事件处理程序。

pin51.com 拼吾爱

Webville TV的一个小内容“计划”

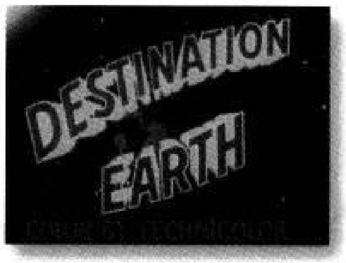
到目前为止，我们只是在Webville TV上运行了一个视频。我们真正想要的是一个节目时间表，来作为视频播放列表。我们希望在Webville TV上做到：



❶ 向观众显示一个小预告片，你懂的，比如可乐和爆米花广告、观众须知等……



❷ 显示我们的第一部正片，《Are you Popular》，相信我们，你肯定会喜欢的。



❸ 然后展示我们的特色影片《Destination Earth》，全彩放映，由American Petroleum Institute出品，他们到底会留下什么消息？仔细看，请找出来。

Sharpen your pencil

现在你可能想看看<video>标记规范来了解如何指定一个播放列表，不过要实现这一点，你需要编写代码，因为<video>元素只允许指定一个视频。如果你孤立无援，只能用浏览器、<video>元素、src属性、load和play方法，以及ended事件实现一个播放列表，你将如何实现（可以使用你喜欢的任何JavaScript数据类型）：

提示：一个视频要结束并停止播放时就会发生ended事件。与其他事件一样，发生这个事件时可以调用一个事件处理程序。

先别偷看答案！

www.pin51.com 拼吾爱

Sharpen your pencil Solution

现在你可能想看看<video>标记规范来了解如何指定一个播放列表，不过要实现这一点，你需要编写代码，因为<video>元素只允许指定一个视频。如果你孤立无援，只能用浏览器、<video>元素、src属性、load和play方法，以及ended事件实现一个播放列表，你将如何实现（可以使用你喜欢的任何JavaScript数据类型）？下面给出我们的答案：

页面加载时，建立一个playlist数组，开始播放第一个视频，并建立一个事件处理程序在这个视频播放结束时调用。

播放列表的代码

创建playlist视频数组

从DOM得到video

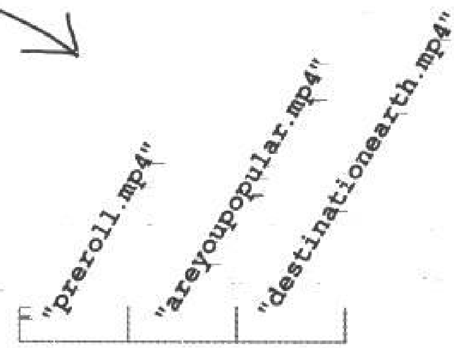
为video的“ended”事件建立事件处理程序

创建变量position = 0

将video源设置为playlist数组中0位置上的视频

播放这个视频

我们会这样存放播放列表，存储为一个数组。每个元素是一个要播放的视频。



Playlist数组

这是处理视频结束事件的处理程序。

Ended事件

每次一个视频结束播放，就会出现ended事件……

……这会调用ended事件处理程序。

Ended事件处理程序的代码

将position增1

设置video为playlist下一个位置上的视频

播放下一个视频

到达播放列表末尾时，可以停止，也可以再循环回去播放第一个视频。

www.pin51.com 拼吾爱 最新最全资源分享下载站

实现Webville TV的播放列表

现在我们要使用JavaScript和视频API来实现Webville TV播放列表。下面先在webvilletv.html中增加一个链接，指向一个JavaScript文件，只需要把它增加到<head>元素中：

```
<script src="webvilletv.js"></script>
```

从现在的<video>元素中删除以下代码：

```
<video controls autoplay src="video/preroll.mp4" width="480" height="360"
       poster="images/prerollposter.jpg" id="video">
</video>
```

（代码中 autoplay src="video/preroll.mp4" 被划掉。）

要从<video>标记删除autoplay和src属性。

还要删除你可能尝试增加的所有<source>元素。

现在，创建一个新文件webvilletv.js，定义一些全局变量，再定义页面完全加载时要调用的一个函数：

```
var position = 0;
var playlist;
var video;
```

首先，定义一个变量来记录我们在播放哪个视频，将这个变量命名为position。

另外需要一个变量保存视频播放列表数组。

还要有一个变量保存video元素的引用。

```
window.onload = function() {
    playlist = ["video/preroll.mp4",
                "video/areyoupopular.mp4",
                "video/destinationearth.mp4"];
    video = document.getElementById("video");
    video.addEventListener("ended", nextVideo, false);

    video.src = playlist[position];
    video.load();
    video.play();
}
```

建立有3个视频的播放列表。

获取video元素。

为视频ended事件增加一个处理程序。对，这看起来与我们以往的做法不同，先别着急，下一页会讨论这个内容。

现在为第一个视频设置src。

加载这个视频，然后播放！

www.pin5i.com 拼吾爱 最新最全下载站 分享资源 激情

事件处理程序代码怎么回事？

以往我们总是把出现一个事件时要调用的处理函数赋给一个属性（如onload或onclick），如下所示：

```
video.onended = nextVideo;
```

不过，这一次我们的做法有点不一样。为什么呢？因为在写这本书时，对视频对象所有事件属性的支持还有一些问题。不过没关系，正是由于存在这个缺陷，我们才有机会向你展示另一种注册事件的方法：addEventListener，这是很多对象都支持的一个通用方法，可以用来注册不同事件。具体做法是这样的：

可以使用addEventListener方法来增加一个事件处理程序。

这是事件发生时将要调用的函数。

```
video.addEventListener("ended", nextVideo, false);
```

我们要在这个对象上监听事件。

这是我们监听的事件。注意在事件名前没有加"on"，这与用属性设置的处理程序不同（如onload）。

第三个参数如果设置为true，它会控制获取事件的一些高级方法。除非你要编写高级代码，否则一般都将这个参数设置为false。

与直接将属性设置为一个函数来增加处理程序相比，addEventListener方法要复杂一些，不过，除此以外，它的做法是一样的。下面再来看我们的代码！

如何编写"视频结束"处理程序

现在只需要为视频的ended事件编写处理程序。只要视频播放器到达当前视频文件的末尾时就会调用这个处理程序。我们可以如下编写这个nextVideo函数（要把它增加到webvilletv.js）：

注意，如果用户暂停视频或者如果视频循环播放（可以设置loop属性来做到），就不会调用这个处理程序。

```
function nextVideo() {
    position++;
    if (position >= playlist.length) {
        position = 0;
    }
    video.src = playlist[position];
    video.load();
    video.play();
}
```

首先，将playlist数组中的position增1。

如果到达播放列表的末尾，会把position设置为0，再循环回到播放列表的第一个视频。

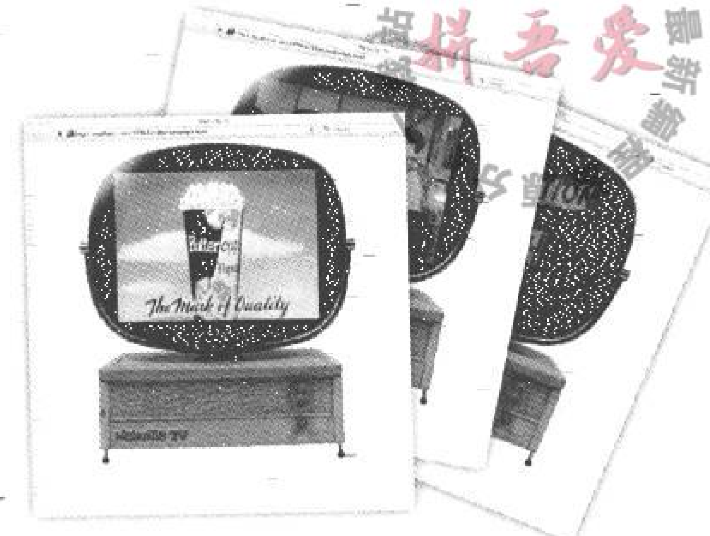
现在将播放器的视频源设置为下一个视频。

最后，加载视频，开始播放这个新视频。

pin51.com 拼吾爱

再来试一试……

你能相信吗？我们居然已经可以试一试了！我们所做的无非是使用API设置一个要播放的视频，然后准备一个事件监听程序来处理视频结束时的情况，它会播放播放列表中的下一个视频。确保对HTML文件完成以上修改，键入新的JavaScript代码，下面来试一试。



这是我们看到的结果，你可以在视频中搓擦（scrub）译注，查看视频切换而不用观看完整的视频。

成功了！不过使用代码加载视频源时如何确定播放哪种视频格式呢？



问得好。

使用多个source标记时，我们可以依赖浏览器检查一个或多个视频格式，并确定是否可以播放其中一种格式。既然我们在使用代码，就只为video元素提供了一个选择。那么如何查看浏览器支持哪种格式来确保我们提供了最佳的格式呢？

要做到这一点，可以使用视频对象的canPlayType方法。canPlayType取一个视频格式为参数，它会返回一个字符串，表示浏览器对于播放这种类型的视频有多大的信心。共有3个信心等级：很可能（probably），可能（maybe）或毫无信心。下面将深入分析，然后会修改播放列表代码来使用这个方法。

你是不是在挠头，喃喃自语"很可能？可能？为什么不直接返回true或false呢？"我们也有同样的疑问，不过稍后再来解释这是什么意思……

译注：搓擦是指用播放头在片段或序列中移动。搓擦用于查找特定的点或帧，或者听音频。

www.pin5i.com 拼吾爱 最新最全的下载站

canPlayType方法如何工作

视频对象提供了一个方法canPlayType，它能确定你能播放一个视频格式的可能性。canPlayType方法取<source>标记中使用的同样的格式描述，返回以下三个值之一：空串、"maybe"或"probably"。可以如下调用canPlayType：

如果只传入一个短形式的格式，只可能得到""或"maybe"作为结果。

```
video.canPlayType("video/ogg")

video.canPlayType('video/ogg; codecs="theora, vorbis"')
```

不过，如果传入带编解码器的具体类型，就可能得到""、"maybe"或"probably"作为答案。

如果浏览器知道无法播放这个视频，则为空串。

如果浏览器认为它有可能播放这个视频，会返回"maybe"。

如果浏览器很有信心，认为能够播放这个视频，就会返回串"probably"。

注意只有当类型中包含了codec参数时浏览器才会有比"maybe"更大的信心。另外要注意，这里没有"我完全确信"之类的返回值。即使浏览器知道它能播放某种类型的视频，仍不能完全保证能够播放具体的视频；例如，如果视频的比特率（bitrate）过高，那么浏览器就可能无法对它解码。

比特率是浏览器每个时间单位必须处理的比特位数，从而能正确地解码和显示视频。

具体使用canPlayType

我们要使用canPlayType来确定Webville TV视频使用哪种视频格式。你已经知道每个文件有3个版本：MP4、WebM和Ogg，取决于你使用哪个浏览器，有些格式可用，有些则不行。下面来创建一个新函数，它会返回适用于你的浏览器的文件扩展名（".mp4"、".webm"或".ogv"）。我们只使用了MIME类型（"video/mp4"、"video/webm"和"video/ogg"），而没有提供编解码器，所以可能的返回值只有"maybe"和空串。代码如下：

```
function getFormatExtension() {
    if (video.canPlayType("video/mp4") != "") {
        return ".mp4";
    } else if (video.canPlayType("video/webm") != "") {
        return ".webm";
    } else if (video.canPlayType("video/ogg") != "") {
        return ".ogv";
    }
}
```

我们知道只可能得到答案"maybe"和空串，所以只需要确保匹配类型不会得到一个空串。

尝试各种类型，如果浏览器表示"我可能支持这种类型"，就返回相应的文件扩展名。

大多数情况下，如果不知道编解码器，能达到"maybe"就已经足够好了。

集成getFormatExtension函数

现在，我们需要对window.onload和nextVideo函数做一些修改来使用getFormatExtension。首先，从播放列表中的文件名删除扩展名（因为我们将使用getFormatExtension来确定文件的扩展名），然后在设置video.src属性时调用getFormatExtension：

```
window.onload = function() {
    playlist = ["video/preroll",
                "video/areyoupopular",
                "video/destinationearth"];
    video = document.getElementById("video");
    video.addEventListener("ended", nextVideo,false);
    video.src = playlist[position] + getFormatExtension();
    video.load();
    video.play();
}
```

删除文件扩展名。现在我们要通过编程来确定文件的扩展名。

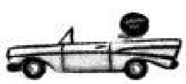
将getFormatExtension的结果与新的视频src文件名连接。

在nextVideo中做同样的处理：

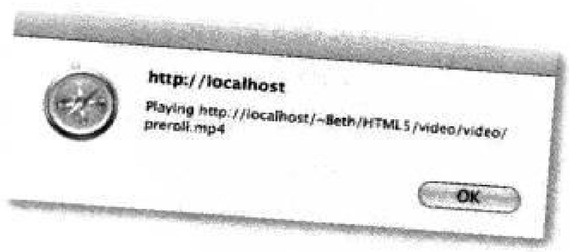
```
function nextVideo() {
    position++;
    if (position >= playlist.length) {
        position = 0;
    }
    video.src = playlist[position] + getFormatExtension();
    video.load();
    video.play();
}
```

这里做同样的处理：将getFormatExtension的结果连接到视频的src。

试一试……



增加canPlayType函数，并完成以上的修改，然后重新加载webvilletv.html文件。能正常工作吗？现在你的代码能确定最佳的格式。如果你想知道浏览器选择了哪个视频，可以尝试在window.onload和nextVideo中增加一个提醒：把它增加到各个函数的最后，也就是video.play()的后面：




```
alert("Playing " + video.currentSrc);
```

你的浏览器播放了哪个文件？

there are no Dumb Questions

问：如果我通过编程设置了视频的源，而且canPlayType的结果是一个“maybe”，但是播放时还是失败了，该怎么处理这种情况呢？

答： 解决这个问题有两种方法。一种方法是捕获错误，并为视频对象指定另一个源（我们将在本章最后讨论如何捕获错误）。另一种方法就是使用DOM在视频对象中一次编写多个source标记（就好像在标记代码中键入一样）。这样一来，你的视频对象就有了多种选择，而你不必编写更多复杂的错误处理代码）。我们不打算在这一章采用这种方法，不过，这确实是为视频对象提供多种选择的一种方法，而且这是通过代码来完成而不是通过标记，如果你有一些高级需求，就可以考虑这种方法。

Watch it!

你可能需要安装Quicktime才能在Safari中播放mp4视频。

Quicktime通常会默认安装，不过如果没有安装，可以从http://www.apple.com/quicktime/download/下载。

Watch it!

Google Chrome有额外的安全限制。

由于有这些安全限制，如果你将Web页面作为一个文件加载（也就是说，你的URL会显示file://而不是http://），将无法完成视频+画布操作，本章后面我们就会使用视频+画布操作。如果你尝试这么做，应用将无法正常工作，而你得不到任何提示来告诉你原因。

所以，对于这一章，我们建议你要么使用另外一个浏览器，要么运行你自己的服务器并从http://localhost运行这些例子。

Watch it!

确保你的服务器提供有正确MIME类型的视频文件。

不论你在使用自己的本地服务器，还是从托管服务器运行一个使用了视频的应用，都需要确保正确地提供视频。如果没有做到，它们可能无法正常工作。

如果你在使用一个本地服务器，系统为Mac或Linux，很可能会使用Apache。可以修改httpd.conf文件（如果有root访问权限），或者在存储视频文件的目录中创建一个.htaccess文件，并增加下面几行：

```
AddType video/ogg  .ogv
AddType video/mp4  .mp4
AddType video/webm  .webm
```

这会告诉服务器如何提供有这些扩展名的文件。

可以在Windows上安装Apache，并做同样的处理。对于IIS服务器，建议你参考Microsoft联机文档“Configuring MIME types in IIS”（IIS中配置MIME类型）。

pin51.com 拼吾爱 最新最全资源分享下载站



有些时候我们必须把你当作一个真正的开发人员看待。

这本书中，我们（真心希望）能帮助你走好每一步。在你跌倒之前，我们已经在前面为你探路，确保在你的代码中不会有基本错误。不过，要成为一个真正的开发人员，很大一部分在于你自己真正深入，阅读其他人的代码，不会只见树木不见森林，并能应对和处理将所有内容集成在一起的复杂性。

在这一章余下的部分中，我们将开始让你做这些工作。接下来，我们会有一个例子，这与我们目前看到的真实Web应用非常接近，它包括很多部分，使用了大量API，还有很多代码来处理大量真实的细节。现在，我们不能再一步一步带着你完成每一部分，像从前一样解释每一个细微差别（否则这本书会有1200页之多）。而且我们也不想这么做，因为你也需要脱离我们，独立地掌握将所有这些部分集成在一起的能力。

不用担心，我们一直在你身边，会告诉你每一部分做什么，但是我们希望你能开始学习如何得到代码，阅读代码，搞清楚，然后对代码做出改进和修改，来实现你希望它完成的工作。所以在接下来的三章中，我们希望你深入学习这些例子，仔细研究，让这些代码植根在你的脑海里。真的…… 准备出发吧！

www.51zxw.net 拼吾爱 最新最全最精的资源分享下载站

我们需要你的帮助！

正好……我们刚拿到合同，要为他们的摄像亭构建一个“**Starring You Video**”软件。这到底是什么？哦，这就是最新的支持HTML5的视频消息传送间，顾客走进一个封闭的摄像亭，录制自己的视频消息，然后可以使用真实的影片效果改进他们的视频。这里有一个老式西部片褐色滤光器，一个黑白电影暗调滤光器，甚至还有一个超现实科幻电影外星滤光器。然后顾客可以把他们的消息发送给朋友。我们已经尽我们所能构建了视频界面和效果处理系统。

不过还存在一个问题。这些摄像亭还有6个星期交付使用，一旦他们交工，你的代码就必须完成。所以，目前我们要有一个已实现部分功能的演示样机，还要有一些测试用的视频文件，我们要在这些基础上编写所有代码。等我们完成任务，Starring You公司的人员就可以用这些代码处理刚采集的实际视频。当然，要记住所有这些都必须用HTML5完成。

所以我们希望你能加入，因为我们签了合同！

走进来，录个视频，加些样式，发给你的朋友！

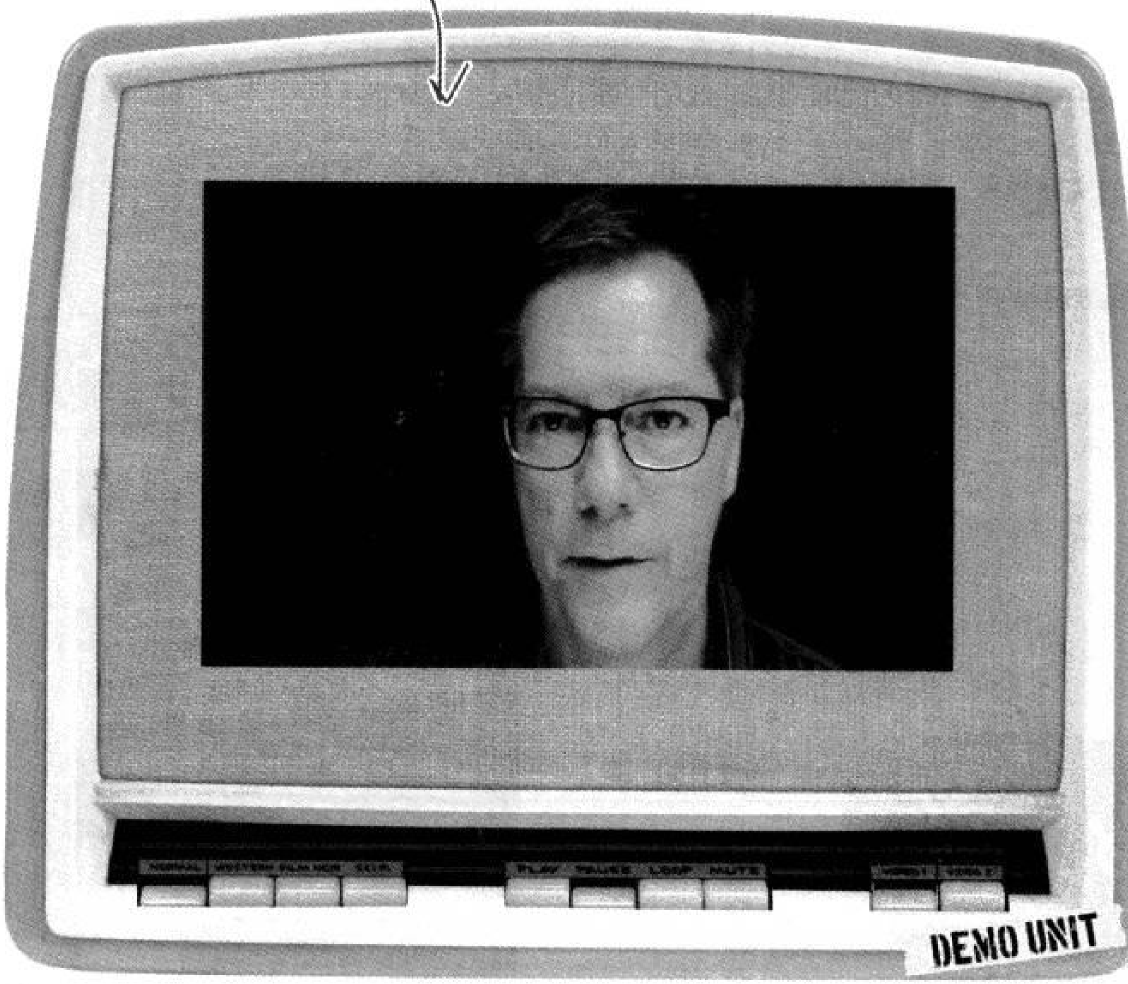


pin51.com 拼吾爱

走进摄像亭，来看一看……

下面来看看我们的演示样机，它有一个用户界面。可以看到一个视频屏幕，用户可以在这里看到播放他们的视频。用户还能应用一个滤光器，如“老式西部片”或“科幻片”滤光器，看看效果怎么样，把它发送给朋友。我们还没有提供记录功能，所以需要一些可以播放的测试视频。我们的第一个任务是把所有内容集成起来，让这些按钮能够正常工作，然后编写视频滤光器。在具体做这些工作之前，先来查看界面：

这是演示样机的界面。在中间有一个视频播放器，可以观看视频。



应用你最喜欢的效果：老式西部片（褐色滤光器）、黑白片（暗调滤光器）或科幻片（反转视频）。

播放、暂停、循环播放和静音控件。

选择一个测试视频。我们的样机有两个测试视频可供选择。



打开演示样机的包装

演示样机已经通过翌日派递送达，现在来打开包装。看起来我们得到了一个已经有基本功能的样机，目前为止已经编写了一些简单的HTML标记和JavaScript。下面先来看看HTML（videobooth.html）。先坐下来，我们有好几页工厂代码要查看，然后就会着手处理真正的代码了。

```
<!doctype html>
<html lang="en">
<head>
   <title>Starring YOU Video Booth</title>
   <meta charset="utf-8">
   <link rel="stylesheet" href="videobooth.css">
   <script src="videobooth.js"></script>
</head>
<body>
<div id="booth">
   <div id="console">
      <div id="videoDiv">
         <video id="video" width="720" height="480"></video>
      </div>
      <div id="dashboard">
         <div id="effects">
            <a class="effect" id="normal"></a>
            <a class="effect" id="western"></a>
            <a class="effect" id="noir"></a>
            <a class="effect" id="scifi"></a>
         </div>
         <div id="controls">
            <a class="control" id="play"></a>
            <a class="control" id="pause"></a>
            <a class="control" id="loop"></a>
            <a class="control" id="mute"></a>
         </div>
         <div id="videoSelection">
            <a class="videoSelection" id="video1"></a>
            <a class="videoSelection" id="video2"></a>
         </div>
   </div>
</div>
</body>
</html>
```

当然是HTML5。

所有样式工作已经替我们完成了！这里是CSS文件。

这里是JavaScript文件，我们需要编写这个文件中的大部分代码。后面会详细介绍这个文件，不过目前看来他们已经编写了控制界面按钮的代码……

这是主界面，我们有了一个控制台，看起来它划分为视频显示区和一个操纵板。操纵板上有3组按钮，分别归组为“effects”（效果）、“controls”（控件）和“videoSelection”（视频选择）。

他们已经安装了一个视频播放器……不错，我们就需要这样一个播放器。

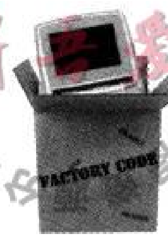
这里是所有效果。

这些都只是HTML锚。稍后会介绍如何关联这些锚……

播放器控件。

两个测试用的演示视频。

检查其余工厂代码



下面来查看从工厂发来的所有JavaScript代码，包括建立按钮的代码（刚才已经在HTML中看到）和各个按钮处理程序的代码（目前，只需确保按下了正确的按钮）。开始增加我们自己的代码之前，先来仔细研究这些现有的代码。

现在来看JavaScript……

下面来分析JavaScript（videobooth.js）。看起来界面上的所有按钮都能正常工作，只不过它们还不能做任何有意思的事情。但重要的是，我们要了解这些按钮是如何建立的，因为这些按钮将调用我们所要写的代码（比如，要播放一个视频或者用一个效果滤光器观看一个视频）。

下面你会看到页面加载时调用的函数。以下代码会逐步处理每一组按钮（效果、控件和视频选择），为锚链接指定点击处理程序。下面就来看一看：

```
window.onload = function() {
    var controlLinks = document.querySelectorAll("a.control");
    for (var i = 0; i < controlLinks.length; i++) {
         controlLinks[i].onclick = handleControl;
    }

    var effectLinks = document.querySelectorAll("a.effect");
    for (var i = 0; i < effectLinks.length; i++) {
        effectLinks[i].onclick = setEffect;
    }

    var videoLinks = document.querySelectorAll("a.videoSelection");
    for (var i = 0; i < videoLinks.length; i++) {
        videoLinks[i].onclick = setVideo;
    }

    pushUnpushButtons("video1", []);
    pushUnpushButtons("normal", []);
}
```

这是页面完全加载时要调用的函数。

每个for语句循环处理一组按钮中的元素。

播放器控件中各个按钮的onclick处理程序设置为handleControl。

效果组中按钮的处理程序设置为setEffect。

最后，视频选择组中按钮的处理程序设置为setVideo。

一旦完成所有基础工作，我们使用一个辅助函数让界面中的“video1”按钮和“normal”（无滤光器）按钮看起来已经按下。

你以前没有见过document.querySelectorAll，它类似于document.getElementsByTagName，只不过你要选择与一个CSS选择器匹配的元素。这个方法会返回一个与CSS选择器参数匹配的元素对象数组。

```
var elementArray = document.querySelectorAll("selector");
```

www.pin5i.com 最新最全 拼吾爱 下载站

查看按钮处理程序

OK，到目前为止，JavaScript代码负责建立所有按钮，从而在点击按钮时会调用适当的处理程序。接下来，我们再来看具体的处理程序，先从播放器按钮的处理程序开始，包括play（播放）、pause（暂停）、loop（循环）和mute（静音），看看它们做些什么：

这个处理程序中首先要做的是查看谁调用了这个处理程序，为此要获取调用这个处理程序的元素的id。

```
function handleControl(e) {
    var id = e.target.getAttribute("id");

    if (id == "play") {
        pushUnpushButtons("play", ["pause"]);

    } else if (id == "pause") {
        pushUnpushButtons("pause", ["play"]);

    } else if (id == "loop") {
        if (isButtonPushed("loop")) {
            pushUnpushButtons("", ["loop"]);
        } else {
            pushUnpushButtons("loop", []);
        }
    } else if (id == "mute") {
        if (isButtonPushed("mute")) {
            pushUnpushButtons("", ["mute"]);
        } else {
            pushUnpushButtons("mute", []);
        }
    }
}
```

一旦知道了id，，可以知道这个元素是play、pause、loop还是mute。

取决于具体的按钮，我们会调整界面来反映按下了哪个按钮。例如，如果按下了pause，play就不能按下。

我们使用了一个辅助函数来负责处理屏幕上按钮的状态，这个函数名为pushUnpushButtons，它有两个参数，一个是按下的按钮，另一个是未按下按钮的数组，它会更新界面来反映这个状态。

不同按钮有不同的语义。例如，play和pause按钮就像真正的单选钮（按下一个，就要弹起另一个），而mute和loop则像是开关按钮。

目前为止所有代码都与界面有关，它只是将按钮的外观从“按下”改为“未按下”。现在还没有代码来完成具体的工作，如播放一个视频。这些代码要由我们来编写。

这确实很不错，不过我们的代码放在哪里呢？下面来考虑一下：当一个按钮（比如说play）被按下，我们不仅要更新界面（前面的代码已经做到了），我们还要增加一些代码来具体做些什么，如让视频开始播放。下面就来看看另外两个处理程序（用来设置视频效果和设置测试视频），从中可以很明显地看出我们的代码该放在哪里……

setEffect和setVideo处理程序

下面来看另外两个处理程序。setEffect处理程序会处理你选择的效果，如无效果（normal）、西部片（western）、黑白片（film noir）或科幻片（sci-fi）。类似地，setVideo处理程序会处理你选择的测试视频，是视频1还是视频2。这两个处理程序的代码如下：

它与handleControl处理程序的做法是一样的：首先获取调用这个处理程序的元素的id（所点击的按钮），然后相应地更新界面。

```
function setEffect(e) {
    var id = e.target.getAttribute("id");

    if (id == "normal") {
        pushUnpushButtons("normal", ["western", "noir", "scifi"]);

    } else if (id == "western") {
        pushUnpushButtons("western", ["normal", "noir", "scifi"]);

    } else if (id == "noir") {
        pushUnpushButtons("noir", ["normal", "western", "scifi"]);

    } else if (id == "scifi") {
        pushUnpushButtons("scifi", ["normal", "western", "noir"]);

    }
}
```

我们的代码要放在这里。

我们将为要处理的各种情况增加代码，来实现相应的特效滤光器。

setVideo也是一样：要查看按下了哪个按钮，并更新界面。

```
function setVideo(e) {
    var id = e.target.getAttribute("id");
    if (id == "video1") {
        pushUnpushButtons("video1", ["video2"]);
    } else if (id == "video2") {
        pushUnpushButtons("video2", ["video1"]);
    }

}
```

还要在这里增加代码，来实现在两个测试视频之间切换。

这里给出辅助函数

为确保完整（或者如果你正搭乘航程11个小时的航班飞往富士山，不能上网，所以确实想键入所有这些代码），下面给出辅助函数的代码：

另外要记住，如果你不想自己键入代码，可以从http://wickedlysmart.com/hfhtml5得到所有代码。



pushUnpushButtons负责按钮状态。它的参数包括一个要按下的按钮的id，以及不再按下的一个或多个按钮（放在一个数组中）。

```
function pushUnpushButtons(idToPush, idArrayToUnpush) {
    if (idToPush != "") {
        var anchor = document.getElementById(idToPush);
        var theClass = anchor.getAttribute("class");
        if (!theClass.indexOf("selected") >= 0) {
            theClass = theClass + " selected";
            anchor.setAttribute("class", theClass);
            var newImage = "url(images/" + idToPush + "pressed.png)";
            anchor.style.backgroundImage = newImage;
        }
    }

    for (var i = 0; i < idArrayToUnpush.length; i++) {
        anchor = document.getElementById(idArrayToUnpush[i]);
        theClass = anchor.getAttribute("class");
        if (theClass.indexOf("selected") >= 0) {
            theClass = theClass.replace("selected", "");
            anchor.setAttribute("class", theClass);
            anchor.style.backgroundImage = "";
        }
    }
}
```

- 首先，查看要按下的按钮的id，确保它不为空。
- 获取使用这个id的锚元素……
- ……得到class属性。
- 在锚中增加“selected”类，从而“按下”这个按钮，并……
- ……更新锚元素的背景图像，用一个“按钮已按下”图像覆盖这个按钮。所以，“pause”按钮会使用“pausepressed.png”图像。
- 要让按钮不再按下，循环处理不再按下的按钮id数组中的各个元素，获取各个锚……
- ……确保这个按钮确实已经按下（如果是，会有一个“selected”类）……
- ……从类中删除“selected”……
- ……另外删除背景图像，从而能看到未按下的按钮。

isButtonPushed只是查看一个按钮是否已经按下。它取一个锚的id……

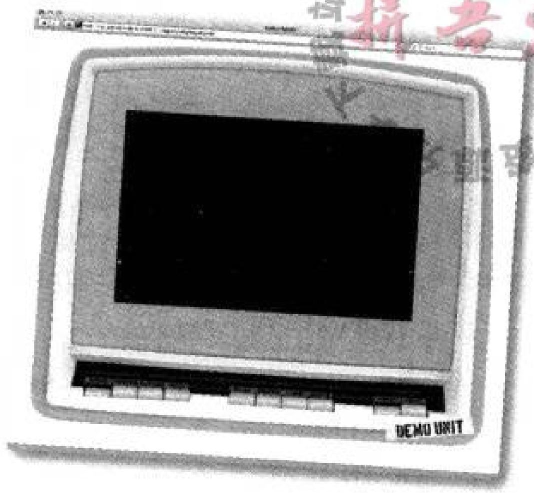
```
function isButtonPushed(id) {
    var anchor = document.getElementById(id);
    var theClass = anchor.getAttribute("class");
    return (theClass.indexOf("selected") >= 0);
}
```

- ……获取这个锚……
- ……得到这个锚的类……
- ……如果这个锚有“selected”类则返回true。

pin51.com 拼吾爱

该试试这个新的演示样机了！

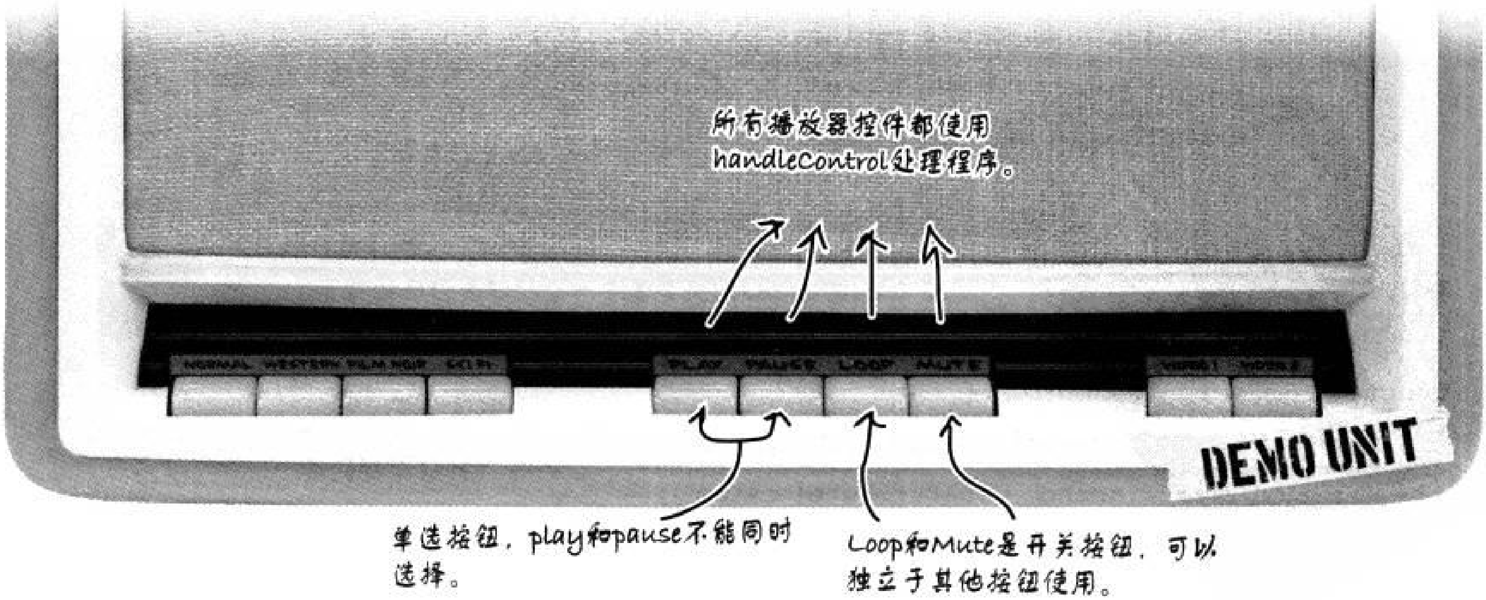
我们还没有编写太多代码，不过一直在阅读和理解代码，这也很不错，收获不小。下面把videobooth.html文件加载到你的浏览器，检查这些按钮。对它们做充分的测试。另外，还可以在处理函数中增加一些提醒，从而更好地理解这些代码的工作。等你完成了测试，我们就会开始编写一些代码来让这些按钮真正起作用。



Sharpen your pencil

再过两页你就会看到这个练习的答案……

对于下面的按钮，请标出它们像开关按钮（独立的按钮）还是像单选钮（按下一个按钮，其他按钮就会弹起来）。还要标出每个按钮相应的点击处理程序。我们已经为你完成了几项：



拼吾爱



这正是CSS的强大之处。

这本书不是《Head First HTML5 Programming with JavaScript & CSS》，真是很遗憾，不过倘若真是那样，这本书就会有1400页了，不是吗？当然了，可能以后会有人让我们写一本高级CSS的书……

不过，说正经的，这正是标记负责结构，而CSS负责表现的威力（如果这对你来说是一个新内容，可以看看《Head First HTML with CSS & XHTML》）。我们做的并没有那么复杂，如果你很好奇，确实想知道，下面做个简单说明：

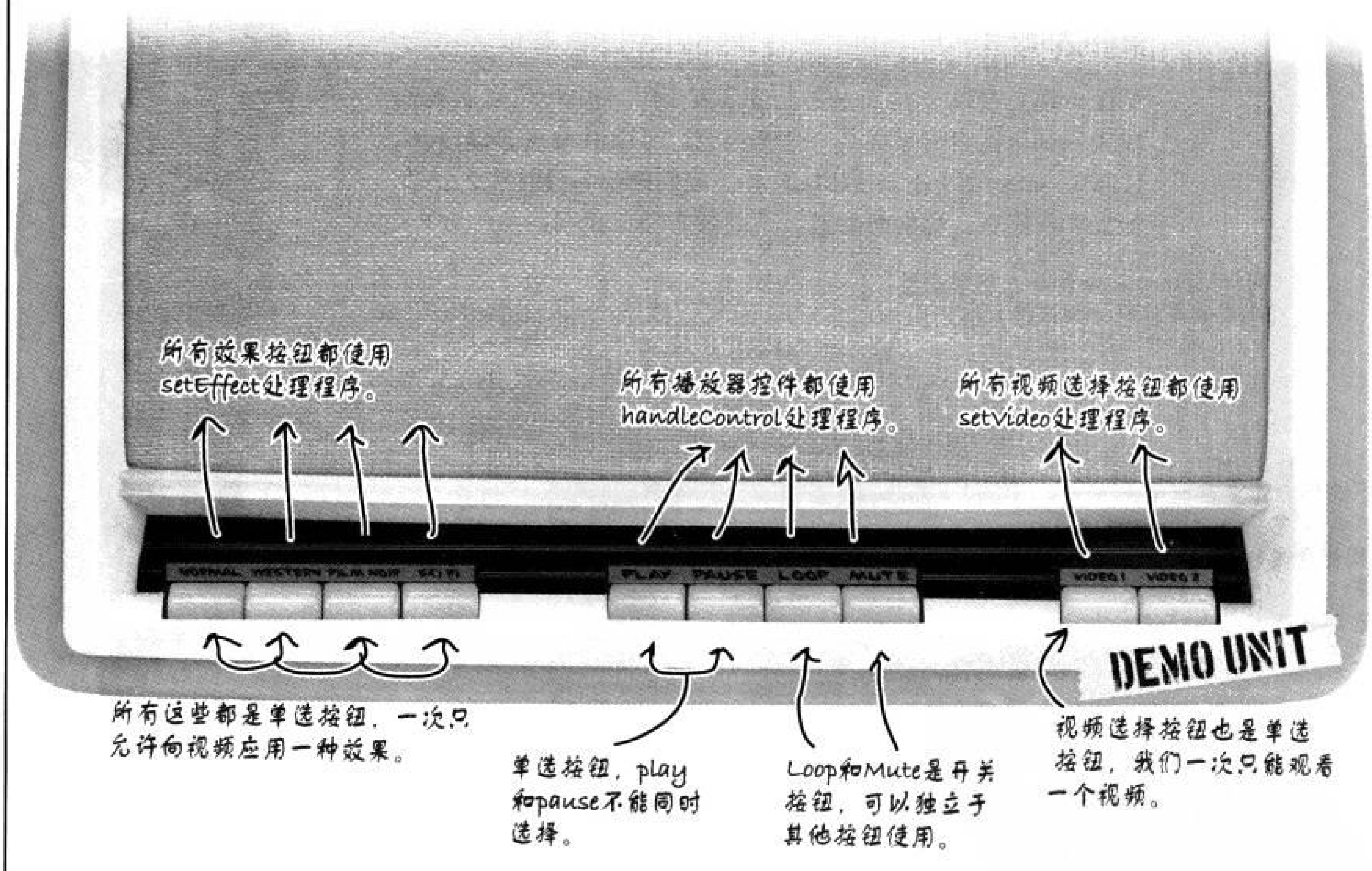


如果想详细查看CSS，可以看看videobooth.css。

pin51.com 拼吾爱 最新最全的资源下载站

Sharpen your pencil Solution

对于下面的按钮，请标出它们像开关按钮（独立的按钮），还是像单选按钮（按下一个按钮，其他按钮就会弹起来）。还要标出每个按钮相应的点击处理程序。以下是我们的答案：



最新下载站 拼吾爱

准备好我们的演示视频……

实现按钮控件之前，还需要有视频能够用来测试，从按钮可以看到，Starring You Video已经给我们发来了两个演示视频。下面就来创建一个对象包含这两个视频，然后为onload处理程序增加一些代码，设置这个视频对象的源（就像我们在Webville TV中所做的一样）。

创建这个对象来包含两个演示视频。稍后还会回来做进一步解释。

```
var videos = {video1: "video/demovideo1", video2: "video/demovideo2"};

window.onload = function() {

    var video = document.getElementById("video");
    video.src = videos.video1 + getFormatExtension();
    video.load();

    var controlLinks = document.querySelectorAll("a.control");
    for (var i = 0; i < controlLinks.length; i++) {
        controlLinks[i].onclick = handleControl;
    }
    var effectLinks = document.querySelectorAll("a.effect");
    for (var i = 0; i < effectLinks.length; i++) {
        effectLinks[i].onclick = setEffect;
    }
    var videoLinks = document.querySelectorAll("a.videoSelection");
    for (var i = 0; i < videoLinks.length; i++) {
        videoLinks[i].onclick = setVideo;
    }
    pushUnpushButtons("video1", []);
    pushUnpushButtons("normal", []);
}
```

这里我们要得到video元素，将它的源设置为数组中第一个有可播放扩展名的视频。

然后加载视频，如果用户点击了play，可以做好准备正常播放视频。

请认真读这段话！

在你松一口气之前，要记住，`getFormatExtension`函数还在Webville TV代码中，而不是在这个代码中！所以，要打开webvilletv.js，把这个函数复制粘贴到你的摄像亭代码中。还有一个小问题要注意，在摄像亭代码中，我们不能保留一个全局视频对象，所以作为弥补，要把下面这行代码增加到`getFormatExtension`函数的最上面：

```
var video = document.getElementById("video");
```

在getFormatExtension函数的最上面增加这行代码。

实现视频控件



现在来实现所有这些按钮。

好了，该实现这些按钮了！现在需要指出重要的一点，对于这个项目，我们要实现我们自己的视频控件。也就是说，并不是使用内置的视频控件，我们要自己来控制应用体验。用户需要播放、暂停视频或者让视频静音，甚至循环播放时，他们会使用我们的定制按钮，而不是使用内置的控件。这也意味着，我们要通过API编程实现所有这些功能。现在我们并不打算一切从零开始，否则这意味着要实现我们自己的视频搓擦条，或者可能还要实现“下一个”和“上一个”按钮，因为这些在这个应用中意义不大，不过如果确实需要也是可以做到的。你会发现，通过实现这个小控制面板，你能很好地掌握有关内容，为进一步深入做好充分准备。

下面就开始吧。先从play按钮开始，然后再向右逐个完成其余按钮（pause，然后是loop，再到mute），怎么样？找到handleControl处理程序，增加以下代码：

```
function handleControl(e) {
    var id = e.target.getAttribute("id");
    var video = document.getElementById("video");

    if (id == "play") {
        pushUnpushButtons("play", ["pause"]);
        if (video.ended) {
            video.load();
        }
        video.play();
    } else if (id == "pause") {
        pushUnpushButtons("pause", ["play"]);

    } else if (id == "loop") {
        if (isButtonPushed("loop")) {
            pushUnpushButtons("", ["loop"]);
        } else {
            pushUnpushButtons("loop", []);
        }
    } else if (id == "mute") {
        if (isButtonPushed("mute")) {
            pushUnpushButtons("", ["mute"]);
        } else {
            pushUnpushButtons("mute", []);
        }
    }
}
```

我们需要视频对象的一个引用。

这应该相当简单。如果用户按下了play，就调用视频对象的play方法。

但是要警告你，这里还有一种边界情况可能会带来问题，所以一定要当心：如果播放一个视频，并让这个视频播放结束，然后再重新开始播放，则必须首先重新加载。所以我们要检查以确保视频确实播放结束（而不是暂停），因为我们只想在结束播放的情况下才重新加载。如果只是暂停，可以直接播放而不需要加载。

拼吾爱 最新 www.pin5i.com 下载站 分享 资源 最新

实现其余视频控件

下面来实现其余的控件。它们都非常简单，简直是小菜一碟：

```
function handleControl(e) {
    var id = e.target.getAttribute("id");
    var video = document.getElementById("video");

    if (id == "play") {
        pushUnpushButtons("play", ["pause"]);
        video.load();
        video.play();
    } else if (id == "pause") {
        pushUnpushButtons("pause", ["play"]);
        video.pause();

    } else if (id == "loop") {
        if (isButtonPushed("loop")) {
            pushUnpushButtons("", ["loop"]);
        } else {
            pushUnpushButtons("loop", []);
        }
        video.loop = !video.loop;

    } else if (id == "mute") {
        if (isButtonPushed("mute")) {
            pushUnpushButtons("", ["mute"]);
        } else {
            pushUnpushButtons("mute", []);
        }
        video.muted = !video.muted;

    }
}
```

如果用户暂停视频，使用视频对象的pause方法。

对于循环播放，视频对象中有一个名为loop的布尔属性。只需正确地设置这个属性……

……为此，你要使用逻辑操作符“!”（非操作符）随时调整，这会将布尔值取反。

mute也采用同样的做法：只需在按下这个按钮时将mute属性的当前值取反。

再来试一试！





确保键入了以上所有修改。在浏览器中加载videobooth.html，测试一下你的控件按钮。应该能看到视频开始播放，你能将它暂停，将它静音，甚至可以让它循环播放。当然，你还不能选择另一个演示视频或者增加一个效果，不过接下来就会做这个工作！

pin51.com 拼吾爱

注意未了结的一个问题……

还有一个未了结的小问题需要注意，这样才能真正让这些按钮完成它们应该完成的工作。下面给出一个用例：你在播放一个视频，而且没有选择loop，这个视频会播放到结束，然后停止。按照我们现在所实现的，play按钮会保持在按下位置。如果它能弹回来，做好准备再次按下是不是更好些？



我们的play按钮还不是尽善尽美……

这一点使用事件可以很容易地做到。下面首先为ended事件增加一个监听程序。将这个代码增加到onload处理程序的最后：

```
video.addEventListener("ended", endedHandler, false);
```

为"ended"事件设置一个处理程序，视频播放结束时就会调用这个处理程序（但是暂停时不会调用）。

现在来编写处理程序，只要达到视频末尾停止播放时就会调用这个处理程序：

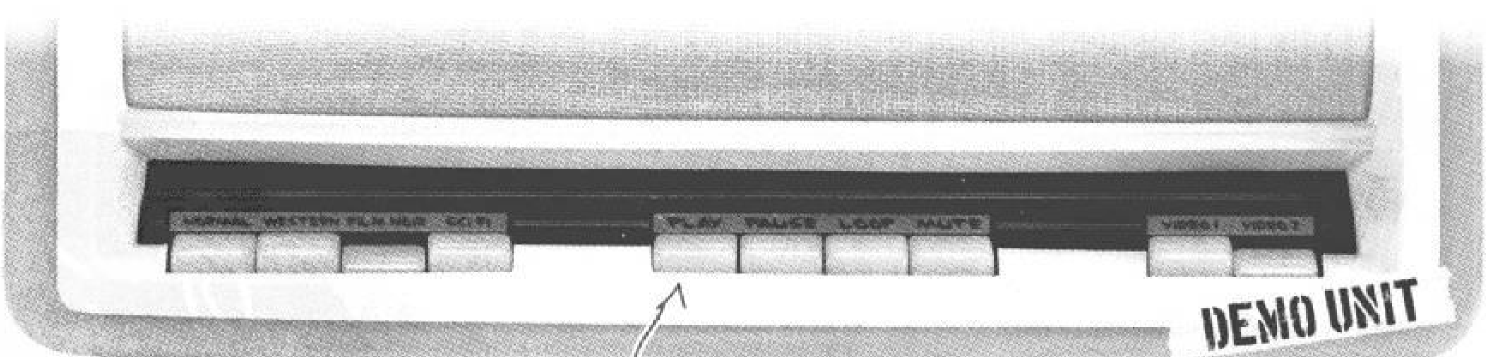
```
function endedHandler() {
    pushUnpushButtons("", ["play"]);
}
```

我们所要做的就是"不再按下"play按钮。就这么简单！

再试一试……



好了，完成以上修改，保存代码，然后重新加载。现在开始播放一个视频，让它播放到结束而不要按下loop按钮，到最后你应该会看到play按钮弹回到它原来的状态。



视频播放结束时play按钮应该弹回到原来的状态。

pan51.com 拼吾爱

切换测试视频



我们已经增加了一个对象来存放两个测试视频，甚至还设置了两个按钮在它们之间做出选择。每个按钮都指定了setVideo处理程序。下面就来编写这个处理程序，从而可以在视频之间进行切换：

以下是包含两个视频的对象，这里再次给出这个对象作为提醒，以便你了解如何使用这个对象……

```
var videos = {video1: "video/demovideo1", video2: "video/demovideo2"};

function setVideo(e) {
    var id = e.target.getAttribute("id");
    var video = document.getElementById("video");

    if (id == "video1") {
        pushUnpushButtons("video1", ["video2"]);
    } else if (id == "video2") {
        pushUnpushButtons("video2", ["video1"]);
    }
    video.src = videos[id] + getFormatExtension();
    video.load();
    video.play();

    pushUnpushButtons("play", ["pause"]);
}
```

这里再次给出处理程序。

同样的，我们需要视频对象的一个引用。

然后仍以同样的方式更新按钮，这里未做修改。

接下来使用按钮的源id（锚的id属性）获取正确的视频文件名，并加上浏览器认识的扩展名。注意我们对videos对象使用了[]记法，这里使用id串作为属性名。

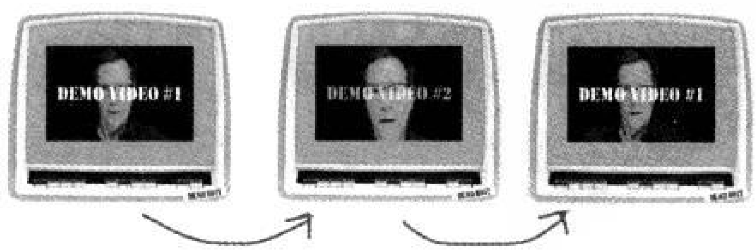
一旦有了正确的视频路径和文件名，可以加载并播放这个视频了。

确保play按钮按下，因为用户点击一个新的视频时，会开始播放这个视频。

切换并测试！



对你的setVideo函数做以上修改，然后再次加载页面。现在应该能轻松地在视频源之间切换了。



pin5i.com 拼吾爱 最新最全资源分享下载站



HTML5机密

本周访谈：
Video元素的告白

Head First: 欢迎你，Video。我正要谈到每个人都很关注的一个话题，那就是关于你和Canvas元素。

Video: 什么意思？

Head First: 现在大街小巷都已经传得沸沸扬扬了，还需要我多说吗？

Video: 有什么可说呢？Canvas和我关系很好呗。她能用一种，怎么说呢，一种看起来很吸引人的方式显示她的内容，而我是一个视频达人。我能处理编解码器，把这些视频内容呈送给浏览器。

Head First: 嗯，这不是我想说的，不过你说的我懂了。是不是这样：她很擅长2D显示，而你的优势在于视频显示。那又怎样？真正的联系是什么？

Video: 就像任何关系一样，当你把两样东西放在一起，会得到比这些部分简单相加更多的结果，正所谓1＋1不等于2，此时就会得到些特殊的东西。

Head First: 好的，嗯，你能不能更具体地讲一讲？

Video: 这是个很简单的概念。如果你希望不只是基本的视频播放，还要做些别的。也就是说，如果你希望对视频做些处理，或者定制覆盖，或者同时显示多个视频，你就会用到画布。

Head First: 听上去还不错，不过视频要求大负荷的处理，我的意思是说，这会有大量的数据传送。JavaScript作为一个脚本语言，怎么能做这些具体的工作呢？编写JavaScript代码可不是使用一种本地语言编写代码。

Video: 哦，你肯定会大吃一惊的…… 你没看过JavaScript最新的基准测试吗？它已经很快了，而且一天比一天快。它是这个行业最闪亮的新星，很多问题都已经解决，相当了不起。

Head First: 噢，不过，这是真的吗？

Video: 千真万确。

Head First: 你能给我们举几个使用JavaScript、画布和视频的例子吗？

Video: 当然，你可以实时地处理视频、检查视频的特性，从视频帧获取数据，还可以通过很多方式修改视频数据，比如说…… 旋转、缩放，甚至修改像素。

Head First: 你能一步一步地告诉我们怎么在代码中完成这些工作吗？

Video: 噢，我很快就回来，刚接到画布的一个电话…… 我得赶快走了……

现在来些特效

现在是不是该增加一些电影效果了？我们希望能够对我们原来的视频应用这些效果，如黑白片、西部片，甚至超现实的科幻片效果。不过，如果查看视频API，你找不到任何效果方法，也没有任何方法能够直接增加效果。那么我们怎么增加这些效果呢？

现在花点时间来考虑如何向视频增加效果。不要担心你还不知道如何处理视频，现在只考虑高层设计。



我们希望能够对原来的视频应用黑白片、西部片和科幻片效果。

Starring You Video
设计说明……

使用这个设计说明来画一个图，做出标记，或者为增加视频效果的代码写出伪代码。可以把这当作为一个热身，只是让你的大脑开动起来……

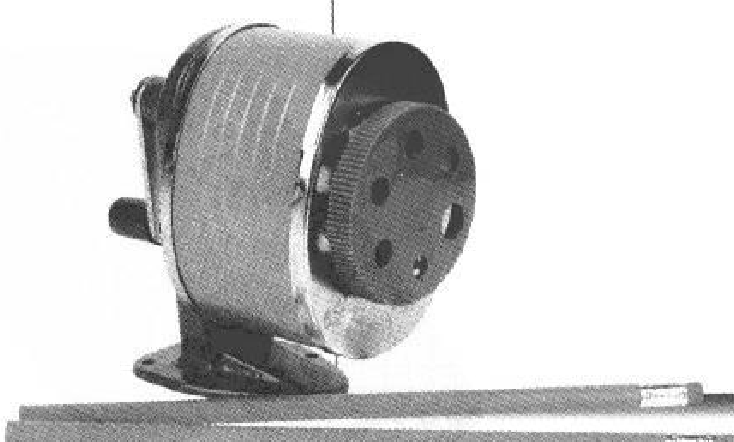
如何得到构成各个视频帧的像素？

一旦得到像素，如何处理这些像素来应用效果？

假设你想编写一个函数来实现各个效果，它会是什么样子？

一旦处理了视频的像素来应用效果，如何显示视频？

把你的想法写在这里。

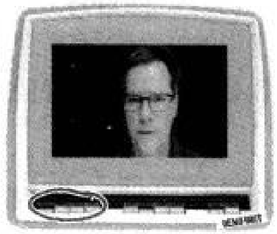


pin5i.com 拼吾爱

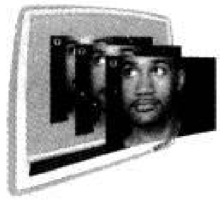
FX计划

对于如何实现这些效果，我们还不是很清楚，不过这里有一个高层计划：

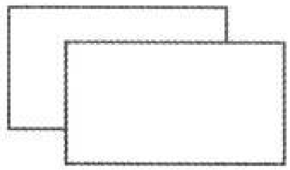
1. 我们知道还必须把这些控制效果的按钮关联起来。所以首先来做这个工作。



2. 我们会学习一些有关视频处理的知识，会利用“scratch缓冲区”技术增加我们的效果。



3. 我们要实现scratch缓冲区，可以利用这个机会来了解视频和画布如何合作。



4. 为每个效果实现一个函数：西部片（western）、黑白片（film noir）和科幻片（sci-fi）。

```
function noir(pos, r, g, b, data) {
   ...
}
```

5. 最后，把所有内容集成在一起，完成测试。



BRAIN POWER

现在你了解了我们将分别实现一个函数来处理各个效果。下面以黑白片为例。如何从视频得到一个有色像素，并把它变成黑白的呢？提示，每个像素有3个分量，分别是红、绿、蓝。如果能得到这些分量，该怎么做呢？

现在让这些效果按钮起作用



好的，首先来完成容易的部分：我们要把这些效果按钮关联起来，让它们起作用。先创建一个全局变量，名为effectFunction。这个变量用来保存一个函数，它能从视频得到数据，并应用一个滤光器。也就是说，取决于我们想要的效果，effectFunction变量会保存一个相应的函数，它知道如何处理视频数据，并使它变成黑白色、褐色，或者反转来实现科幻片效果。所以将这个变量增加到文件的最前面：

```
var effectFunction = null;
```

只要点击一个效果按钮，就会设置这个变量。对于现在，我们将使用western、noir和scifi等函数名，稍后来编写这些函数。

这里再次给出setEffect处理程序。要记住，只要用户点击一个效果按钮就会调用这个处理程序。

按下各个按钮时，我们将把effectFunction变量设置为相应的函数（这些函数还需要我们编写）。

```
function setEffect(e) {
    var id = e.target.getAttribute("id");

    if (id == "normal") {
        pushUnpushButtons("normal", ["western", "noir", "scifi"]);
        effectFunction = null;
    } else if (id == "western") {
        pushUnpushButtons("western", ["normal", "noir", "scifi"]);
        effectFunction = western;
    } else if (id == "noir") {
        pushUnpushButtons("noir", ["normal", "western", "scifi"]);
        effectFunction = noir;
    } else if (id == "scifi") {
        pushUnpushButtons("scifi", ["normal", "western", "noir"]);
        effectFunction = scifi;
    }
}
```

如果效果是“没有效果”或正常（normal），则使用null值。

否则，将effectFunction设置为一个适当命名的函数，它将对视频具体应用效果。

还需要编写这些效果函数。所以下面来看如何处理视频来应用效果！

好了，有了这些基础，下面就来学习“scratch缓冲区”，然后再回来看看如何加入这些函数，以及如何编写这些函数！

pin51.com 拼吾爱

如何完成视频处理

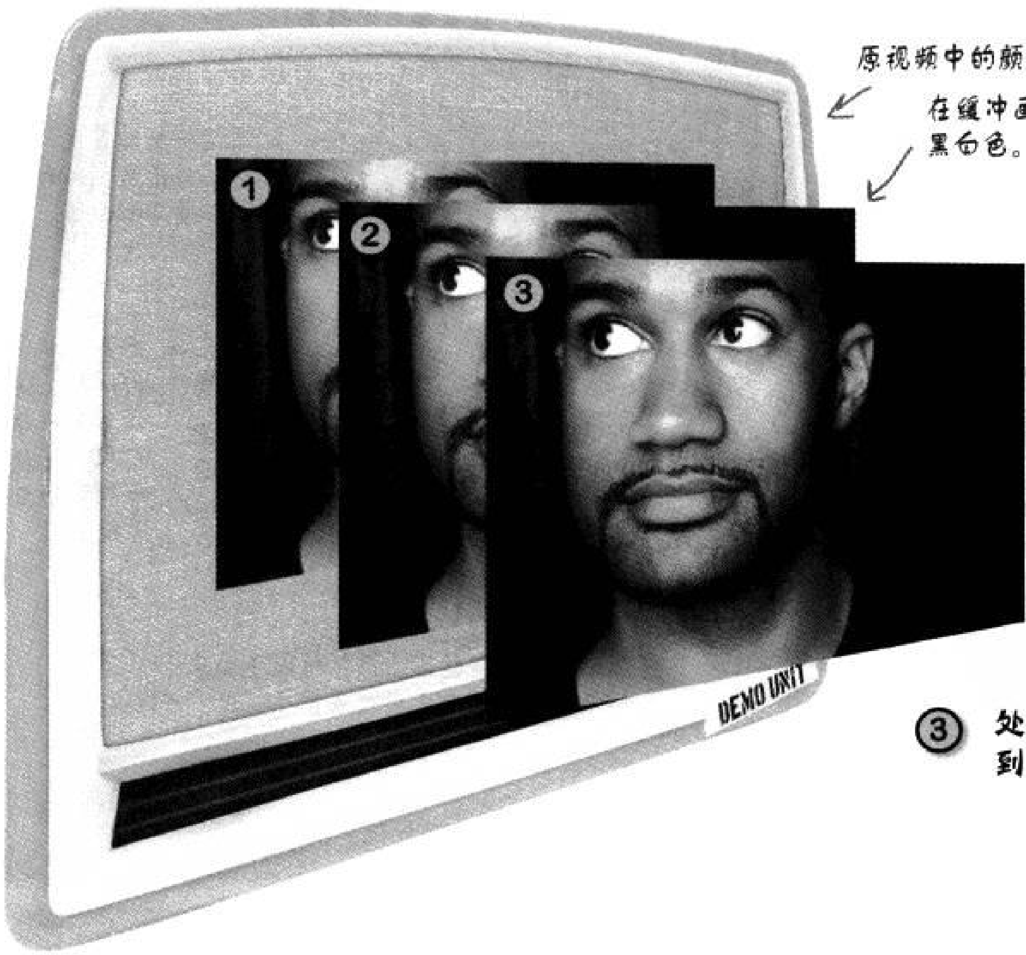
目前为止我们所做的就是，想办法为effectsFunction全局变量指定一个函数，作为点击界面中效果按钮的结果。现在你要掌握这个知识，把它牢牢记住，因为我们要搞清楚如何得到视频并实时处理来增加效果。为了达到这个目的，我们需要得到视频的像素，修改这些像素来实现我们所需的效果，然后以某种方式再把它们放回到屏幕上。

那么，视频API有没有提供一种特殊的方法可以在显示视频之前先处理视频呢？没有。不过它确实提供了一种方法来得到像素，所以我们只需要有办法处理和显示像素就可以了。等一下，像素？显示？还记得第7章吗？画布！哈，完全正确，我们确实提到过video元素和canvas有一种“特殊的关系”。那么，下面就来了解video和canvas元素合作的一种方法：

独家报道的细节，终于揭开了！

① 视频播放器在后台解码，并播放视频。

② 视频逐帧地复制到一个（隐藏的）缓冲画布并处理。



③ 处理一帧之后，将它复制到另一个显示画布来观看。

pin51.com 拼吾爱 最新最全的资源下载站

在幕后

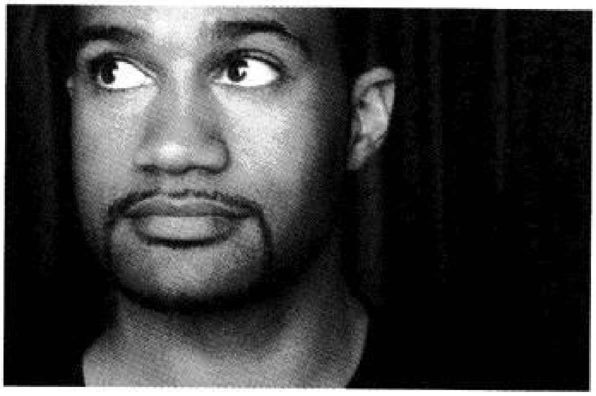
如何使用scratch缓冲区处理视频

现在你可能会问，为什么我们要使用两个画布来处理和显示视频呢？为什么不找个办法在对视频解码的同时就完成处理呢？

这里使用的方法是一种已经得到充分证明的成熟技术，可以在密集的视频和图像处理中尽可能减少视觉抖动，这种技术称为使用“scratch缓冲区”。通过在一个缓冲区中处理一个视频帧，然后把它完全复制到显示画布，可以尽量减少视觉问题。

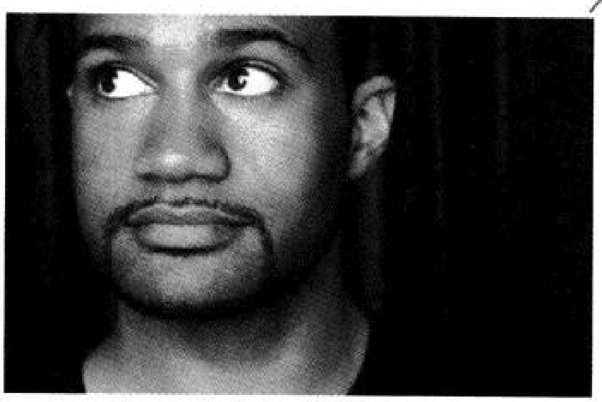
下面来逐步了解scratch缓冲区技术如何工作。

① **浏览器将视频解码为一系列帧。每一帧是一个像素构成的矩形，包含指定时间点的视频快照。**



一个视频帧。

② **解码每一帧视频时，我们将它复制到作为scratch缓冲区的画布。**



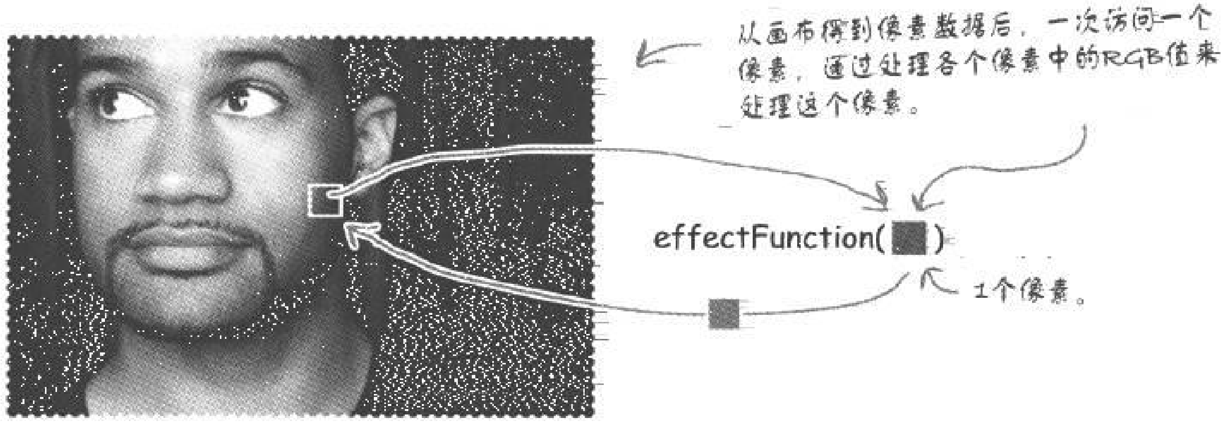


将整个帧复制到画布。

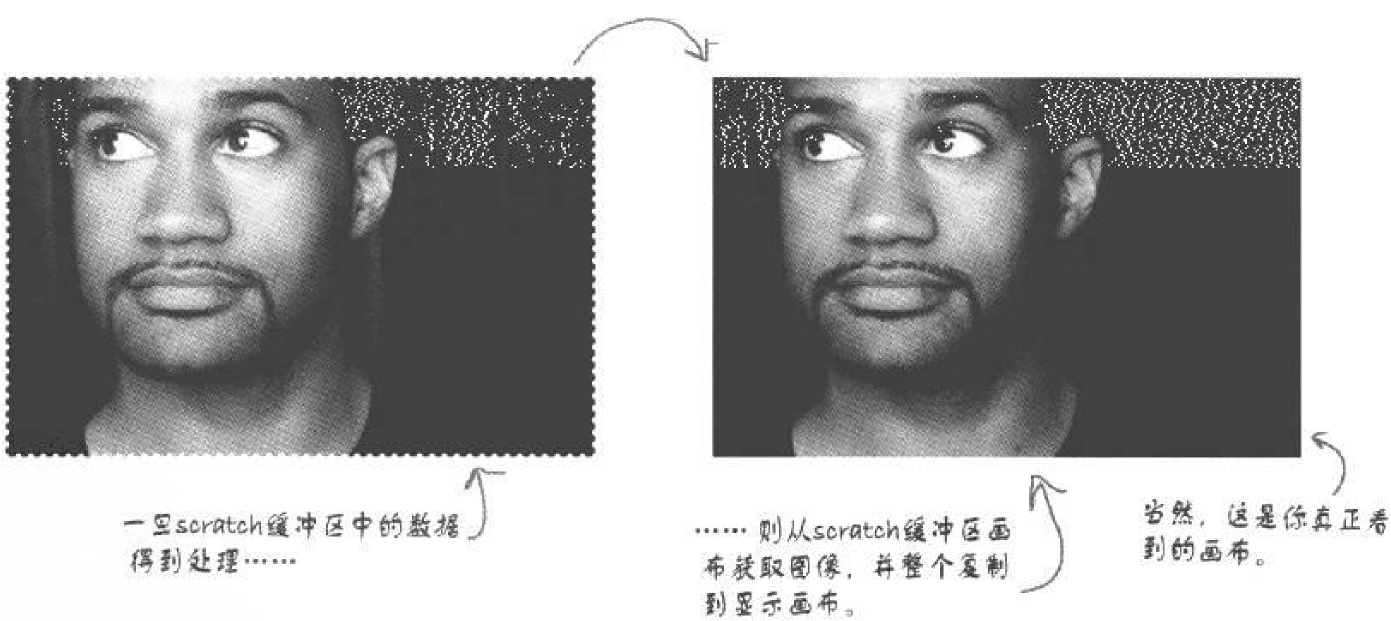
这是scratch缓冲区。

pin51.com 拼吾爱 最新最全资源分享下载站

③ 逐像素地迭代处理scratch缓冲区，将各个像素传入我们的效果函数来进行处理。



④ scratch缓冲区中的所有像素都处理完之后，将它们从scratch缓冲区画布复制到显示画布。



⑤ 由视频对象解码时对每一帧重复这个处理。

www.minxue51.com 拼吾爱 最新最全资源分享下载站

用画布实现scratch缓冲区

你已经知道，要在画布中实现一个scratch缓冲区需要两个画布：一个用来在其中完成计算，另一个用来显示我们的结果。要创建这些画布，需要再回到我们的HTML文件videobooth.html。打开这个文件，找到id为“videoDiv”的<div>，在这个<video>下面增加两个canvas元素：

```
<div id="videoDiv">
    <video id="video" width="720" height="480"></video>
    <canvas id="buffer" width="720" height="480"></canvas>
    <canvas id="display" width="720" height="480"></canvas>
</div>
```

我们要增加两个canvas元素，一个用作缓冲区，另一个用来显示。

注意，它们与video元素大小相等。

如何指定视频和画布的位置

现在你可能想知道这些元素如何定位，我们将让它们相互叠放。最下面是video元素，上面是缓冲区，缓冲区的上面是显示canvas元素。我们将用CSS来实现，尽管这本书里没有太多地谈到CSS，不过如果打开videobooth.css，会看到这3个元素如何定位：

```
div#videoDiv {
    position: relative;
    width: 720px;
    height: 480px;
    top: 180px;
    left: 190px;
}
video {
    background-color: black;
}
div#videoDiv canvas {
    position: absolute;
    top: 0px;
    left: 0px;
}
```

videoDiv <div>相对于它所在的元素（console <div>）放置，距上180px，距左190px，这会将它放在控制台的中央。我们将宽度和高度设置为与<video>以及两个<canvas>元素的宽度和高度相等。

<video>是videoDiv <div>中的第一个元素，所以它会自动地定位在<div>的左上角。我们将背景设置为黑色，这样如果上下加边或左右加边，空白处会是黑色。

videoDiv <div>中的两个<canvas>元素相对于videoDiv（其父元素）定位，采用绝对定位方式，所以将<canvas>元素放在距上0px，距左0px的位置，这样它们就会与<video>和videoDiv在同一个位置上。

pin51.com 拼吾爱 最新最全资源下载站

编写代码处理视频

现在我们有了一个video元素，一个将作为缓冲区的画布，还有一个要显示最终视频帧的画布。我们让它们依次叠放，这样只会看到最上层的显示画布，其中包含应用了效果的视频。为了处理视频，我们将使用video元素的play事件，视频一旦开始播放就会调用这个事件。将下面的代码增加到onload处理程序的最后：

```
video.addEventListener("play", processFrame, false);
```

视频开始播放时，它会调用函数processFrame。

我们将在processFrame函数中处理视频像素，并把处理后的像素放入画布中显示。首先要确保能够访问所有DOM对象：

```
function processFrame() {
    var video = document.getElementById("video");
    if (video.paused || video.ended) {
        return;
    }
    var bufferCanvas = document.getElementById("buffer");
    var displayCanvas = document.getElementById("display");
    var buffer = bufferCanvas.getContext("2d");
    var display = displayCanvas.getContext("2d");
}
```

首先获取video对象……

……查看视频是否仍在播放。如果没有播放，说明没有工作可做，直接返回。

然后获取两个画布元素的引用，以及它们的上下文的引用，我们需要用到这些引用。

如何创建缓冲区

要创建缓冲区，需要得到当前视频帧，将它复制到缓冲区画布。一旦将视频帧放在画布上，就可以处理帧中的数据了。所以，要创建这个缓冲区，我们可以这样做（在processFrame的最后增加以下代码）：

它取一个图像，把这个图像绘制到画布上，使它位于x,y位置，并有指定的宽度和高度。

还记得第7章介绍过的上下文的drawImage方法吗？

这一次我们从视频得到一个图像。将视频指定为源，drawImage会得到一个视频帧作为图像数据。

```
buffer.drawImage(video, 0, 0, bufferCanvas.width, bufferCanvas.height);
var frame = buffer.getImageData(0, 0, bufferCanvas.width, bufferCanvas.height);
```

然后从画布上下文获取图像数据，把它存储在一个变量frame中，以便处理。

这只是表示希望得到画布中的所有图像数据。

mn51.com 最新最全的下载站 拼吾爱

如何处理缓冲区

我们已经得到一帧视频数据，下面对它做些处理！要处理这个视频帧，我们要循环处理帧数据中的每一个像素，取出存储在各个像素中的RGB颜色值。实际上，每个像素有4个值，RGB和Alpha（透明度），不过我们并不打算使用Alpha。一旦得到RGB值，我们将调用effectFunction（应该记得，这是392页上建立的函数，要求你把它牢牢记住的），并提供这个RGB信息和视频帧。

将下面的代码增加到processFrame函数的最后：

```
buffer.drawImage(video, 0, 0, bufferCanvas.width, displayCanvas.height);
var frame = buffer.getImageData(0, 0, bufferCanvas.width, displayCanvas.height);

var length = frame.data.length / 4;

for (var i = 0; i < length; i++) {
    var r = frame.data[i * 4 + 0];
    var g = frame.data[i * 4 + 1];
    var b = frame.data[i * 4 + 2];
    if (effectFunction) {
       effectFunction(i, r, g, b, frame.data);
    }
}
display.putImageData(frame, 0, 0);
```

首先得出帧数据的长度。注意，数据放在帧的一个属性frame.data中，而长度是frame.data的一个属性（length）。这个长度实际上是画布大小的4倍，因为每个像素都有4个值：RGBA。

现在循环处理这个数据，得到每个像素的RGB值。每个像素在数组中占4个空间，所以我们从第一个位置获取r（红色分量），从第二个位置获取g（绿色分量），从第三个位置获取b（蓝色分量）。

然后，调用effectFunction（如果不为null，倘若按下了"Normal"按钮effectFunction就会是null），传入像素的位置、RGB值，以及frame.data数组。效果函数会用新的像素值更新frame.data数组，根据赋至effectFunction的滤光器函数进行处理。

到了这里，帧数据已经得到处理，所以我们使用上下文的putImageData方法将这个数据放在显示画布上。这个方法取帧中的数据，把它写到画布上指定的x,y位置。

已经处理了一帧，下一帧呢？

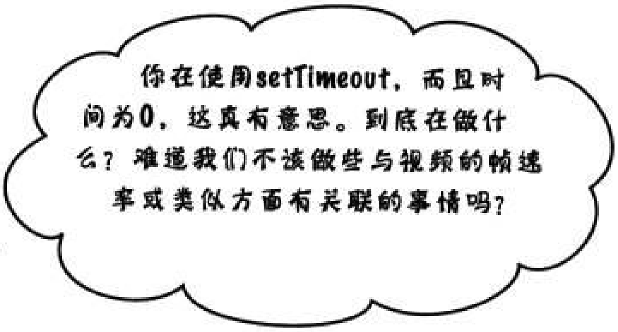
是的，我们刚处理了一帧，希望随着视频的继续播放能够处理所有视频帧。可以使用setTimeout并传入0毫秒值，要求JavaScript尽快再次运行processFrame。JavaScript并不会真的在0毫秒内就运行这个函数，不过它会提供所能得到的下一个最快的时间片。为此，只需要将下面的代码增加到processFrame函数的最后：

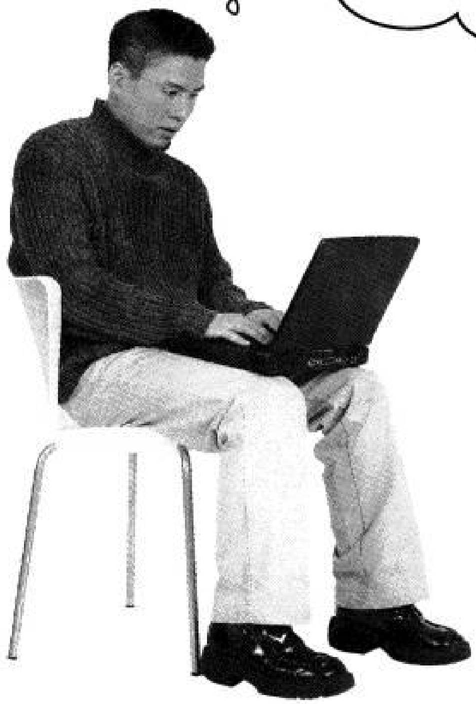
```
setTimeout(processFrame, 0);
```

告诉JavaScript尽快再次运行processFrame。

setTimeout与setInterval类似，只不过它只在指定时间（单位为毫秒）之后运行一次。

pin51.com 拼吾爱





真希望能这样。

你说的完全正确：我们真正想做的是对每一帧调用一次处理程序，但是视频API没有提供这样一种方法。它确实提供了一个名为`timeupdate`的事件，可以用来更新视频的运行时显示，不过它并不按处理帧所用的粒度来更新（换句话说，它的运行速率要低于视频的帧速率）。

所以我们要使用`setTimeout`。向`setTimeout`传入0时，就是在要求JavaScript尽可能快地运行你的超时处理程序，这会使处理程序以尽可能高的频度运行。

不过有没有可能比帧速率还快呢？如果能计算出一个与所需帧速率接近的超时值，会不会更好一些？嗯，当然可以这么做，不过处理程序不太可能真的与视频帧同步运行，所以0是一个很好的近似值。当然，如果你想提高应用的性能，完全可以做一些性能测试，找出最优值。但在得到一个更特定的API之前，我们就打算采用这种方法了。

现在需要写一些效果

终于到这一步了，我们已经得到编写视频效果所需的全部内容：在视频帧到来时获取每一帧，逐像素地访问这一帧，并把像素发送给我们的效果滤光器函数。下面来看暗调滤光器（在我们的这个版本中，这就代表黑白滤光器）：

向滤光器函数传入像素的位置……

……红、绿、蓝像素值……

……以及画布中帧数据数组的一个引用。

```
function noir(pos, r, g, b, data) {
    var brightness = (3*r + 4*g + b) >>> 3;
    if (brightness < 0) brightness = 0;
    data[pos * 4 + 0] = brightness;
    data[pos * 4 + 1] = brightness;
    data[pos * 4 + 2] = brightness;
}
```

所以我们首先要做的是根据这个像素的所有分量（r、b和g）计算一个亮度值。

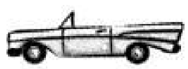
>>>是一个位操作符，可以移动数值中的位来修改这个数。这个内容可以找一本JavaScript参考书进一步研究。

然后将画布图像中的各个分量赋为这个亮度。

要记住，这个函数要为视频帧中每个像素调用一次。

这样做的效果是将像素设置为一个灰度值，对应这个像素的总体亮度。

黑白效果测试



把这个函数增加到videobooth.js，然后重新加载页面。一旦视频开始播放，按下Film Noir（黑白）按钮，你会看到一个郁闷的黑白片。现在再选择Normal（正常），不错吧，是不是？所有功能都用JavaScript完成，而且是实时的！

如果仔细想想，这确实挺惊人的。



www.pin51.com 拼吾爱

Sharpen your pencil

这本书并不是专门来讨论视频处理和效果，不过这个内容确实很有趣。下面我们给出了西部片和科幻片的效果。仔细研究这个代码，在右边做出标注，说明每个代码分别做什么。哦，我们还增加了额外的一个效果，它有什么作用？

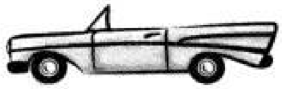
```
function western(pos, r, g, b, data) {
    var brightness = (3*r + 4*g + b) >>> 3;
    data[pos * 4 + 0] = brightness+40;
    data[pos * 4 + 1] = brightness+20;
    data[pos * 4 + 2] = brightness-20;
}

function scifi(pos, r, g, b, data) {
    var offset =  pos * 4;
    data[offset] = Math.round(255 - r) ;
    data[offset+1] = Math.round(255 - g) ;
    data[offset+2] = Math.round(255 - b) ;
}

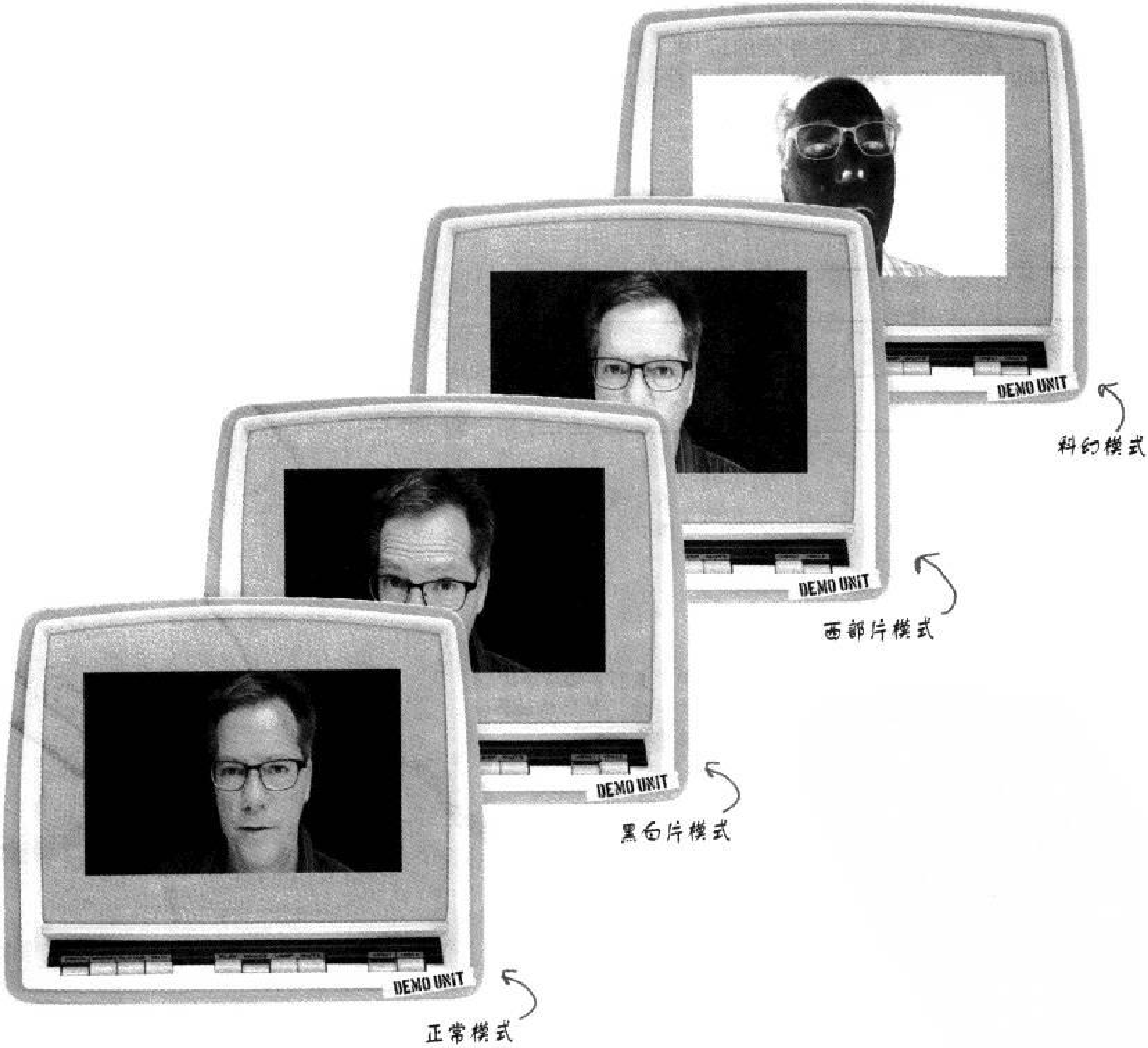
function bwcartoon(pos, r, g, b, outputData) {
    var offset =  pos * 4;
    if( outputData[offset] < 120 ) {
        outputData[offset] = 80;
        outputData[++offset] = 80;
        outputData[++offset] = 80;
    } else {
        outputData[offset] = 255;
        outputData[++offset] = 255;
        outputData[++offset] = 255;
    }
    outputData[++offset] = 255;
    ++offset;
}
```

拼吾爱

大型测试



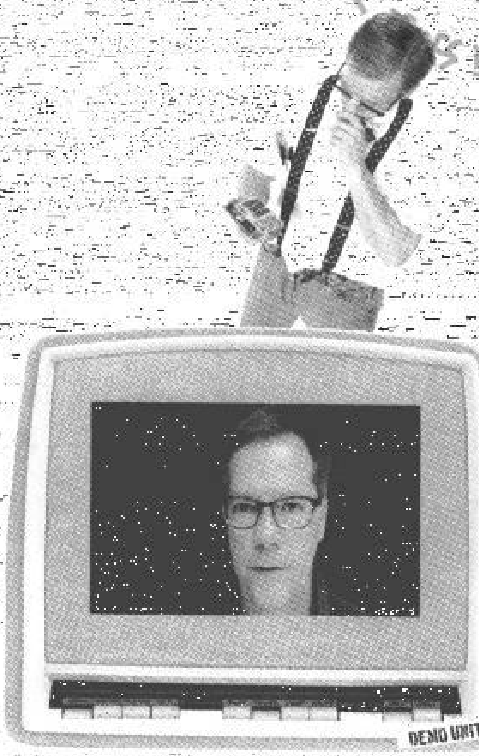
搞定了！把这些代码包装起来，准备交付给**Starring You Video**。再反复检查一下是不是已经输入了所有代码，是否保存，并加载`videobooth.html`。然后你就能用你的新应用好好乐一乐了！



实验室生活

很显然，在视频处理方面我们还只是刚刚接触了一点皮毛，我们相信，你肯定能想出更有创意的效果。你可以大胆去想，努力实现，在这里给出说明。

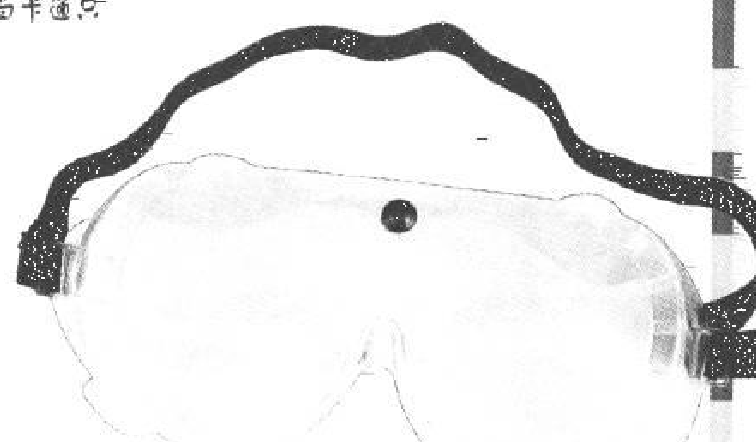
如果你发明并实现了相当酷的应用，请告诉我们（wickedlysmart.com），我们会推广给其他读者！



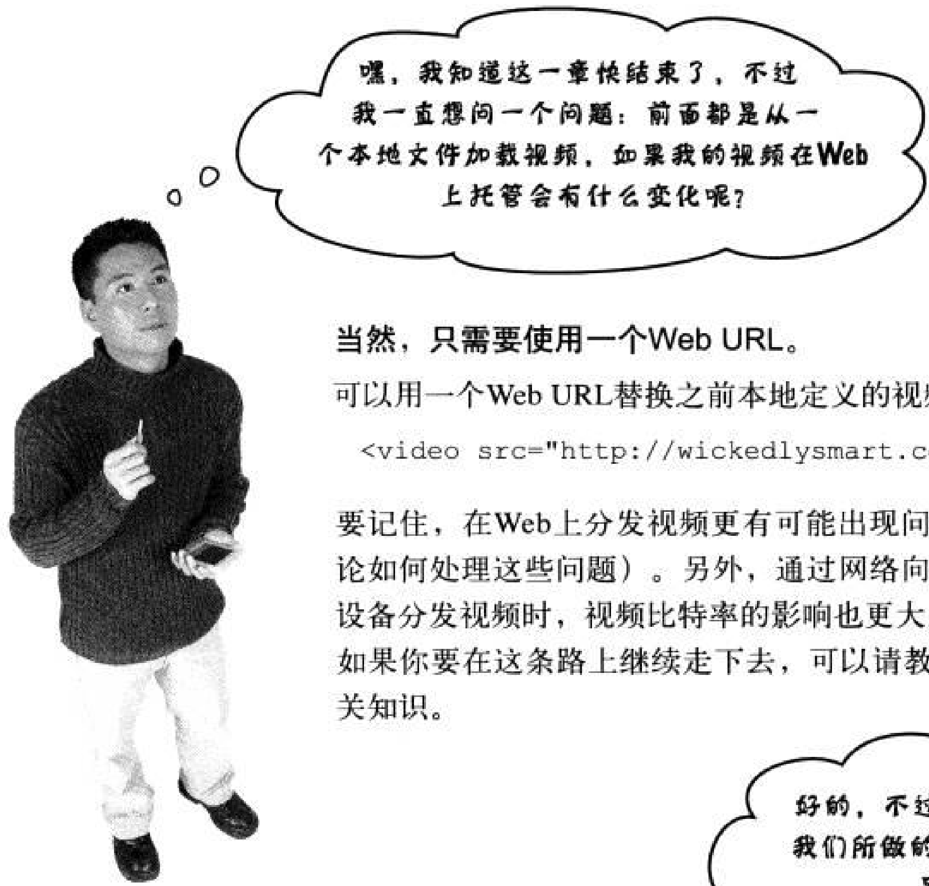
在这里写出你的想法！



利用效果可以做很多有趣的事情，黑白卡通只是其中之一。



pin51.com 拼吾爱 最新资源分享下载站

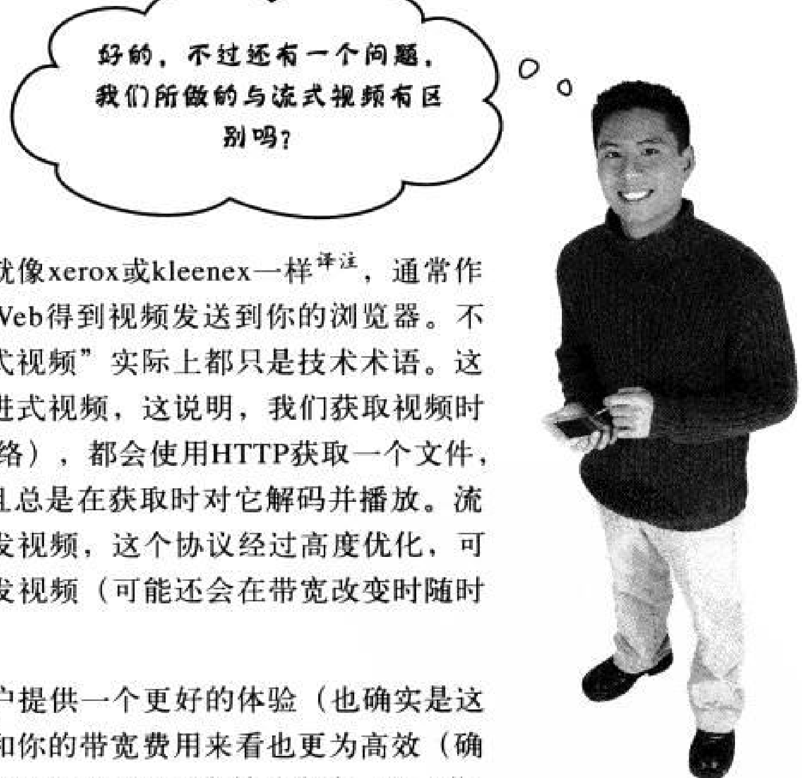


当然，只需要使用一个Web URL。

可以用一个Web URL替换之前本地定义的视频源。例如：

```
<video src="http://wickedlysmart.com/myvideo.mp4">
```

要记住，在Web上分发视频更有可能出现问题（稍后我们会讨论如何处理这些问题）。另外，通过网络向一个浏览器或移动设备分发视频时，视频比特率的影响也更大。类似于视频格式，如果你要在这条路上继续走下去，可以请教专家，了解更多有关知识。



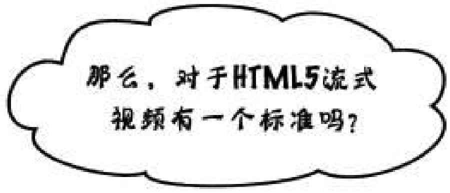
是的，有很大的区别。

流式（streaming）这个词就像xerox或kleenex一样[译注]，通常作为一个通用词汇，表示从Web得到视频发送到你的浏览器。不过“渐进式视频”和“流式视频”实际上都只是技术术语。这本书中我们一直在使用渐进式视频，这说明，我们获取视频时（不论是在本地还是通过网络），都会使用HTTP获取一个文件，如HTML文件或图像，而且总是在获取时对它解码并播放。流式视频则使用一个协议分发视频，这个协议经过高度优化，可以采用一种最优的方式分发视频（可能还会在带宽改变时随时间调整视频的比特率）。

听上去流式视频可以为用户提供一个更好的体验（也确实是这样），而且从用户的连接和你的带宽费用来看也更为高效（确实如此）。不仅如此，流式视频还可以更容易地完成一些工作，比如倘若你需要某种安全性，它能保护你的视频内容。

译注：xerox（施乐）是一家主营打印机、复印机等数字印刷设备的公司，现在已经成为“复印”的代名词。kleenex（舒洁）是全球最知名的面巾纸品牌，也已经成为“面巾纸”的通用名词。

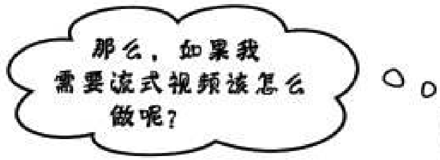
pin51.com 拼吾爱 最新最全资源分享下载站





没有。

HTML5的流式视频并没有一个标准。实际上，问题并非出在HTML5，不论在哪里都没有为流式视频提供一个得到支持的标准—不过确实有很多专用标准。为什么？这有很多原因，从建立流式视频所需的资金，到崇尚开源的很多人不想使用一个可能用于数字版权管理（DRM）或其他保护技术的协议，这些都是原因。与视频格式的情况类似，在流式视频方面我们也面对着一个复杂的世界。





现在有很多解决方案。

流式视频技术有很多合法使用，如果你的观众数量很大，或者有一些你认为需要保护的内容，可以查查看：Apple的HTTP Live Streaming、Microsoft的Smooth Streaming和Adobe的HTTP Dynamic Streaming都是不错的起点。

还有一个好消息要告诉你：标准委员会开始仔细研究基于HTTP的流式视频，所以请关注这个领域的进展。

www.aq51.com 易特书城

如果有个完美的世界……



可惜这个世界并不完美：我们会遇到那些烦人的网络问题、不兼容的设备和操作系统，而且行星撞击地球的可能性也越来越大。最后这个麻烦我们爱莫能助，不过对于前两个，实际上如果能知道遇到一个错误就是成功了一半，这样起码你能对它做些处理。

视频对象有一个error事件，可能会因为很多原因抛出错误，这个原因可以在video.error属性中找到，或者更具体地可以查找video.error.code属性。下面来看可以检测哪些类型的错误：

错误

MEDIA_ERR_ABORTED=1

如果通过网络得到视频的过程被浏览器中止（可能是应用户的请求），就会使用这个错误。

MEDIA_ERR_NETWORK=2

网络获取视频时如果被一个网络错误中断，就会使用这个错误。

MEDIA_ERR_DECODE=3

如果视频解码失败就会使用这个错误。发生这个错误可能是因为编码使用了浏览器不支持的特性，或者因为文件已经破坏。

MEDIA_ERR_SRC_NOT_SUPPORTED=4

如果不支持指定的视频源，可能由于URL有问题，或者由于浏览器无法解码这种源类型，就会使用这个错误。

每个错误类型都有一个相关的编号，这是error事件产生的错误码，稍后还会介绍……

pin51.com 拼吾爱 最新最全资源分享下载站

如何使用error事件

处理错误是一个很复杂的事情，具体如何处理错误很大程度上取决于你的应用，需要适合你的应用和你的用户。也就是说，我们至少可以帮你起步，为你指出正确的方向。下面以Webville TV为例，使它能知道遇到了错误。如果确实遇到了一个错误，会向观众提供一个PLEASE STAND BY消息。

我们希望出现一个错误消息时得到通知，所以需要为error事件增加一个监听程序。可以这样做（将这个代码增加到`webville.js`中的onload处理程序）：

```
video.addEventListener("error", errorHandler, false);
```

出现一个错误时，会调用errorHandler函数。

现在来编写函数`errorHandler`，它要检查是否出现一个错误，如果有，就把我们的“please stand by”图像放在视频显示区上，让它作为海报图像：

如果调用了这个处理程序，首先检查video.error，确认确实存在一个错误，然后在视频显示区放一个海报图像。

```
function errorHandler() {
    var video = document.getElementById("video");
    if (video.error) {
        video.poster = "images/technicaldifficulties.jpg";
        alert(video.error.code);
    }
}
```

这是可选的，可以增加这行代码，这样就能看到错误码（见上一页存储在code属性中的整数）。

错误测试！



可以通过很多方式让视频播放失败，要测试这个代码，必须让它失败。下面给出几个建议：



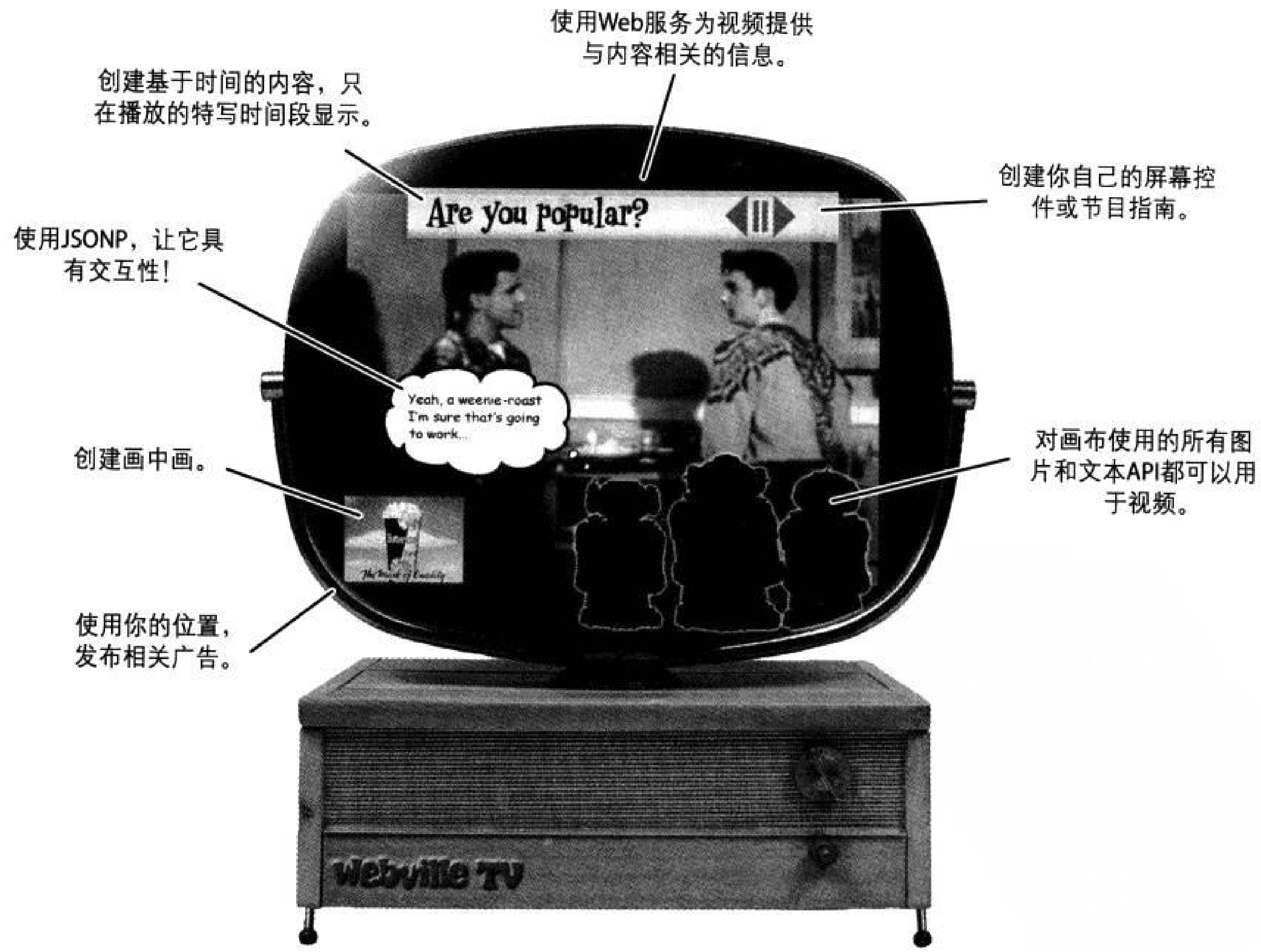
- 在播放过程的不同时间点断开网络连接。
- 为播放器提供一个有错误的URL。
- 为播放器提供一个你知道它无法解码的视频。
- 为播放器提供一个根本不是视频的URL。
- 使用软件减少你的带宽（有很多这样的软件，你可以找找看）。

现在键入这个代码来完成测试。记住，可以查看405页上的代码，把对话框中的整数映射为具体的错误码。

www.pin5i.com 拼吾爱 最新最全源分享下载站

现在还要做什么？

真是让人兴奋，想想看，你现在已经了解如何处理HTML标记，如何处理video元素，以及，当然了，还有画布……更不用说Web服务、地理定位…… 哇。确实，我们用画布完成了一些很酷的视频处理，关于处理画布你所知道的一切都能应用到视频。这里只给出我们的几个想法，你可以天马行空，增加你自己的想法。你可以犒劳犒劳自己了，这是你应得的！



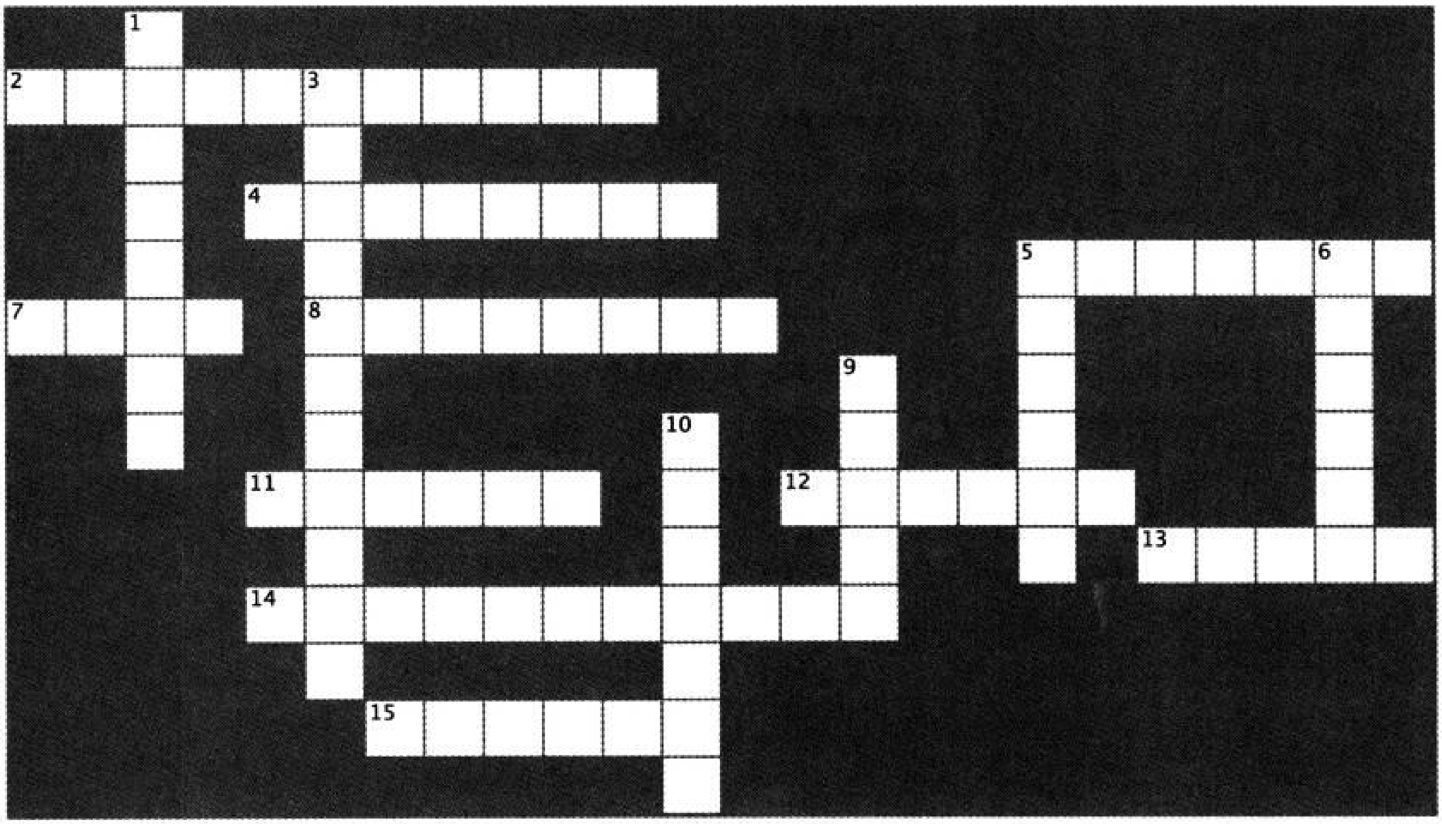
pin5i.com 拼吾爱

BULLET POINTS

- 可以使用<video>元素和一简单的属性播放视频。
- autoplay属性在页面加载时就开始播放，不过只应在适当的情况下使用。
- controls属性会使浏览器提供一组播放控件。
- 不同浏览器提供的控件的外观有所不同。
- 可以用poster属性提供你自己的海报图像。
- src属性包含要播放的视频的URL。
- 对于视频和音频格式有很多“标准”。
- 3种常用的格式包括WebM、MP4/H.264和Ogg/Theora。
- 要了解你的观众，从而知道需要提供哪些格式。
- 使用<source>标记来指定候选的视频格式。
- 在<source>标记中使用完全限定类型可以节省浏览器的工作和时间。
- 可以继续支持其他视频框架，如Flash，只需在video元素中增加一个作为后路的<object>标记。
- 视频对象提供了一组丰富的属性、方法和事件。
- 视频支持播放、暂停、加载、循环和静音方法及属性来直接控制视频的播放。
- 可以利用ended事件了解视频播放何时结束（例如，来实现一个播放列表）。
- 可以用canPlayType通过编程询问视频对象能不能播放某种格式。
- canPlayType方法可能返回空串（不支持这种格式）、maybe（可能可以播放这种格式），或者probably（它对能够播放这种格式很有信心）。
- 画布可以用作为视频的显示表面，来实现定制控件或视频的其他效果。
- 可以使用一个scratch缓冲区，在将视频复制到显示表面之前先对视频进行处理。
- 可以使用setTimeout处理程序来处理视频帧。尽管没有直接链接到视频的每一帧，但这是目前最好的方法。
- 可以使用一个URL作为视频源来播放网络视频。
- 有些浏览器对视频有一个同源策略，要求从源页面同样的来源提供视频。
- 关于视频很有可能会出现错误，特别是涉及网络时。
- 利用error事件，可以在视频获取、解码或播放过程中出现错误时通知一个处理程序。
- video元素依赖于渐进下载的视频。目前对于流式视频还没有HTML5标准，不过标准委员会正在寻求基于HTTP的流式解决方案。
- 目前对于保护通过video元素分发的视频没有标准的方法。

HTML5填字游戏

坐下来观看更多Webville TV节目之前，先来做一个简单的填字游戏，把你学到的知识牢牢记住。下面是第8章的填字游戏。



横向

2. video元素用这种方式分发视频。
4. 要提供多个视频选择，可以使用 ______source元素。
5. 我们使用画布作为 ______ 缓冲区。
7. 用来反复播放视频的属性。
8. 尽快地开始播放一个视频。
11. 开源的音频编解码器。
12. 用来显示处理后的视频。
13. 播放结束时，会抛出这个事件。
14. 我可以播放这种类型，你能吗？
15. 浏览器控件的外观______。

纵向

1. 如果希望采用一种内置的方式控制视频要使用 ____________ 。
3. 我们看过20世纪50年代 _______ 影片。
5. 如果一个行星要撞击地球，你应该怎么做。
6. 星巴克CEO洒了他的____________。
9. 每个setTimeout调用时处理的是什么。
10. 克林特·伊斯特伍德喜欢这种效果。

pin51.com 拼吾爱

Sharpen your pencil Solution

这本书并不是专门来讨论视频处理和效果，不过这个内容确实很有趣。下面我们给出了西部片和科幻片的效果。仔细研究这个代码，在右边做出标注，说明每个代码分别做什么。哦，我们还增加了额外的一个效果，它有什么作用？下面是我们的答案。

```
function western(pos, r, g, b, data) {
    var brightness = (3*r + 4*g + b) >>> 3;
    data[pos * 4 + 0] = brightness+40;
    data[pos * 4 + 1] = brightness+20;
    data[pos * 4 + 2] = brightness-20;
}
```

西部片滤光器会增强像素的红色和绿色分量，而弱化蓝色分量，这会给视频一种棕褐色色彩。

```
function scifi(pos, r, g, b, data) {
    var offset =  pos * 4;
    data[offset] = Math.round(255 - r) ;
    data[offset+1] = Math.round(255 - g) ;
    data[offset+2] = Math.round(255 - b) ;
}
```

科幻片滤光器将各个像素的RGB分量值取反。如果一个像素的红色分量很大，现在就会很小。如果一个像素的绿色分量很少，现在则会很多。

```
function bwcartoon(pos, r, g, b, outputData) {
    var offset =  pos * 4;
    if( outputData[offset] < 120 ) {
        outputData[offset] = 80;
        outputData[++offset] = 80;
        outputData[++offset] = 80;
    } else {
        outputData[offset] = 255;
        outputData[++offset] = 255;
        outputData[++offset] = 255;
    }
    outputData[++offset] = 255;
    ++offset;
}
```

黑白卡通滤光器将每个红色分量小于120（上限255）的像素变为黑色，将所有其他像素变为白色，使视频有一种奇怪的类似卡通的黑白外观。

www.ppzw.com 拼吾爱 最新电子书

TOP SECRET

CASE FILE: VIDEO

视频侦察

答案

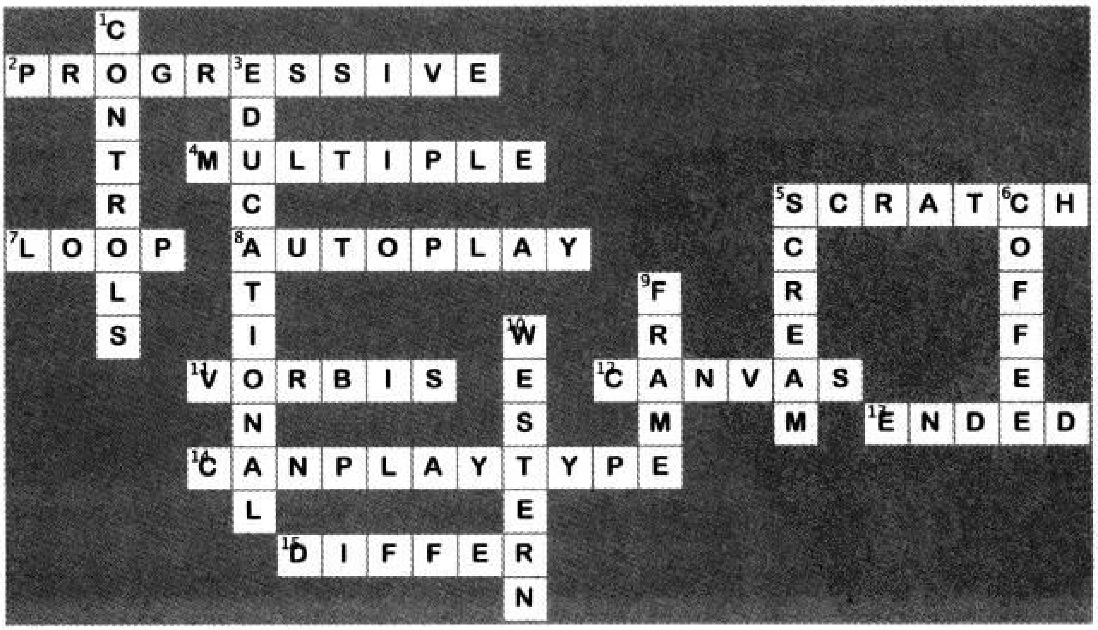
iOS和Android设备（以及其他）→ Mobile Webkit

Video \ Browser	Safari	Chrome	Firefox	Mobile Webkit	Opera	IE9+	IE8	IE7 or <
H.264	✓	部分		✓		✓		
WebM		✓	✓		✓			
Ogg Theora		✓	✓		✓			

Watch it!

这会很快发生变化！所以要查看Web上最新的支持情况。

HTML5填字游戏答案



pin51.com 拼吾爱 最新编程资源分享下载站

9 在本地存储

Web存储



是不是厌倦了把客户数据塞在那个小小的~~衣柜~~cookie里？这样做在20世纪90年代还算有点意义，不过如今的Web应用早已经有了更大的需求。如果我们告诉你每个用户的浏览器上可以提供5M的存储空间，你可能会像看外星人一样看着我们，就好像我们在布鲁克林给你推销一座大桥一样。不过，真的不用怀疑，HTML5 Web Storage API确实可以做到这一点！这一章我们将带你了解在用户设备上本地存储对象所需掌握的全部知识，可以充分用于你的Web体验中。

pin51.com 拼吾爱 最新最全最好的免费分享下载站

pin51.com 拼吧网 最新资源分享站

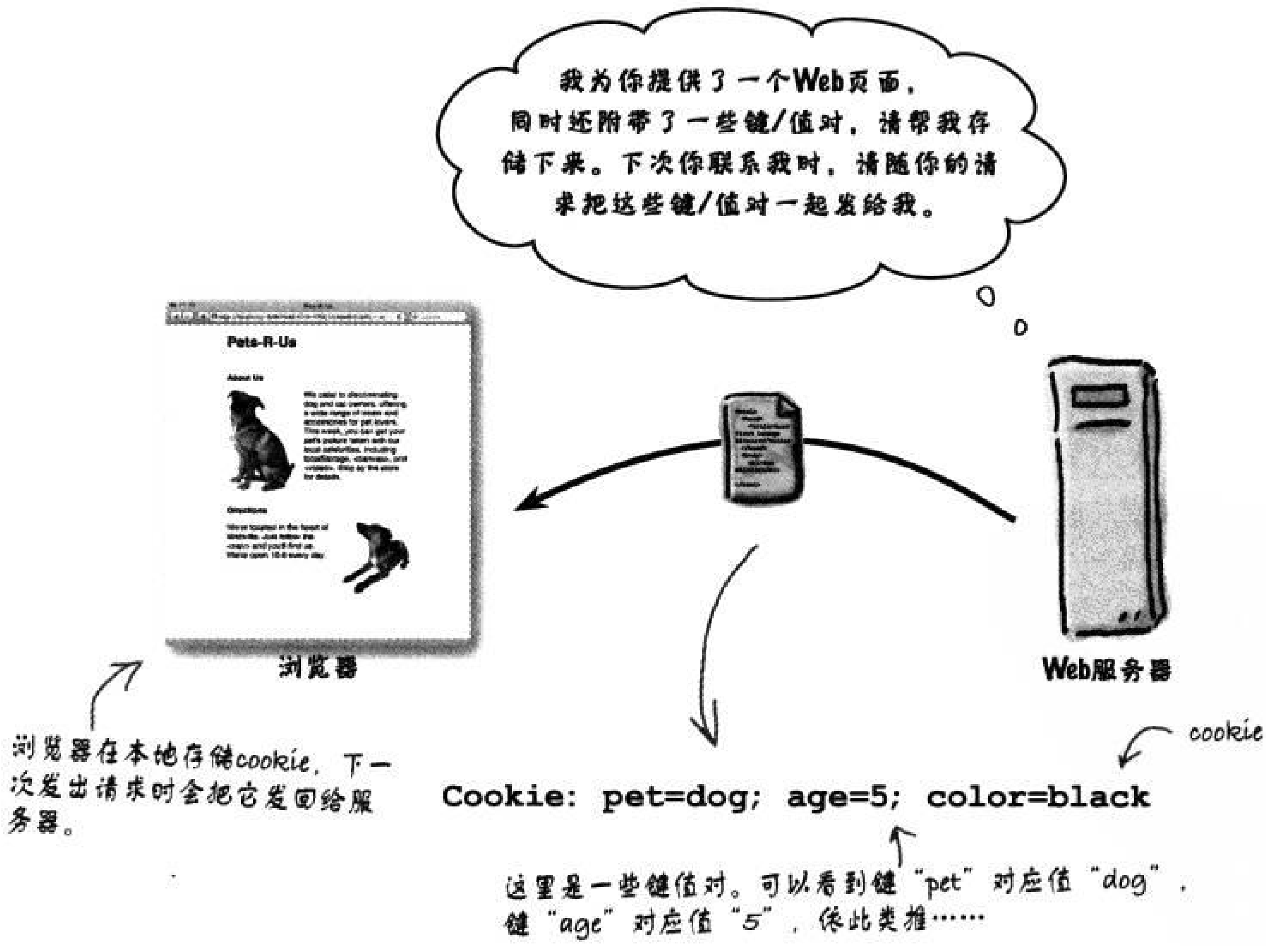
浏览器存储如何工作(1995~2010)

在幕后

是不是想建个购物车？需要为你的网站存储一些用户首选项吗？或者是不是只需要将与各个用户关联的一些数据藏起来？这就引入了浏览器存储。浏览器存储为我们提供了一种持久存储数据的方法，从而在构建Web应用时可以利用存储的这些数据。

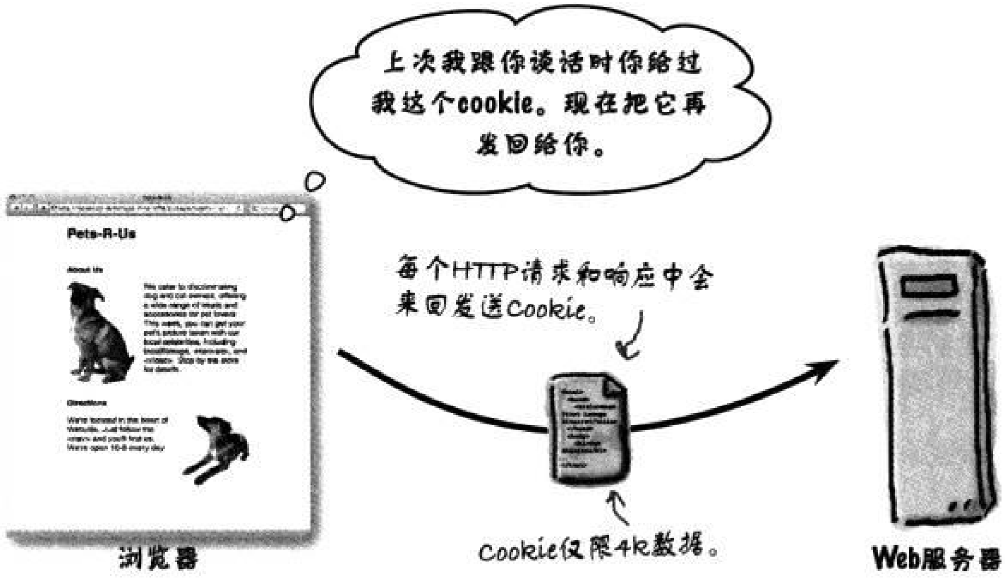
到目前为止，只有一条路可走，这就是利用浏览器cookie在浏览器上存储信息。下面来看cookie是如何工作的：

1. 浏览器获取一个Web页面，比如从“pets-R-us.com”获取页面，服务器可以随响应发送一个cookie。Cookie中包含一个或多个键值对：



pin5i.com 拼吾爱 最新最全最快资源分享下载站

2 下次浏览器向"pets-R-us.com"发出请求时，它会随请求同时发送之前收到的cookie：



3 服务器可以使用cookie实现个性化的体验，在这里就是向用户推荐一些相关商品，不过cookie还有很多其他的用法。



pin51.com 拼吾爱 最新编程资源分享下载站

BRAIN POWER

Cookie陪伴我们了很长时间，不过你可能可以想出一些办法对它们做出改进。

你认为cookie在以下哪些方面存在问题，请打勾：

- [ ] 只有4k空间可以使用，我的应用需要更多的存储空间。
- [ ] 每次都要来回发送cookie，这看起来确实很低效，特别是如果我在使用一个移动设备，它可没有太多带宽。
- [ ] 听起来利用cookie很容易向浏览器传送病毒和其他恶意软件。
- [ ] 我听说如果把键/值对作为HTTP请求的一部分，在代码中处理时会很麻烦。
- [ ] 是不是每次做出请求时都可能来回发送一些私人数据？
- [ ] 看起来cookie不适合我们所做的所有客户端开发，似乎它们假设所有一切都在服务器上完成。

准确地讲，尽管有负面的新闻报道，但实际上cookie很安全，并不是病毒的安全港湾。



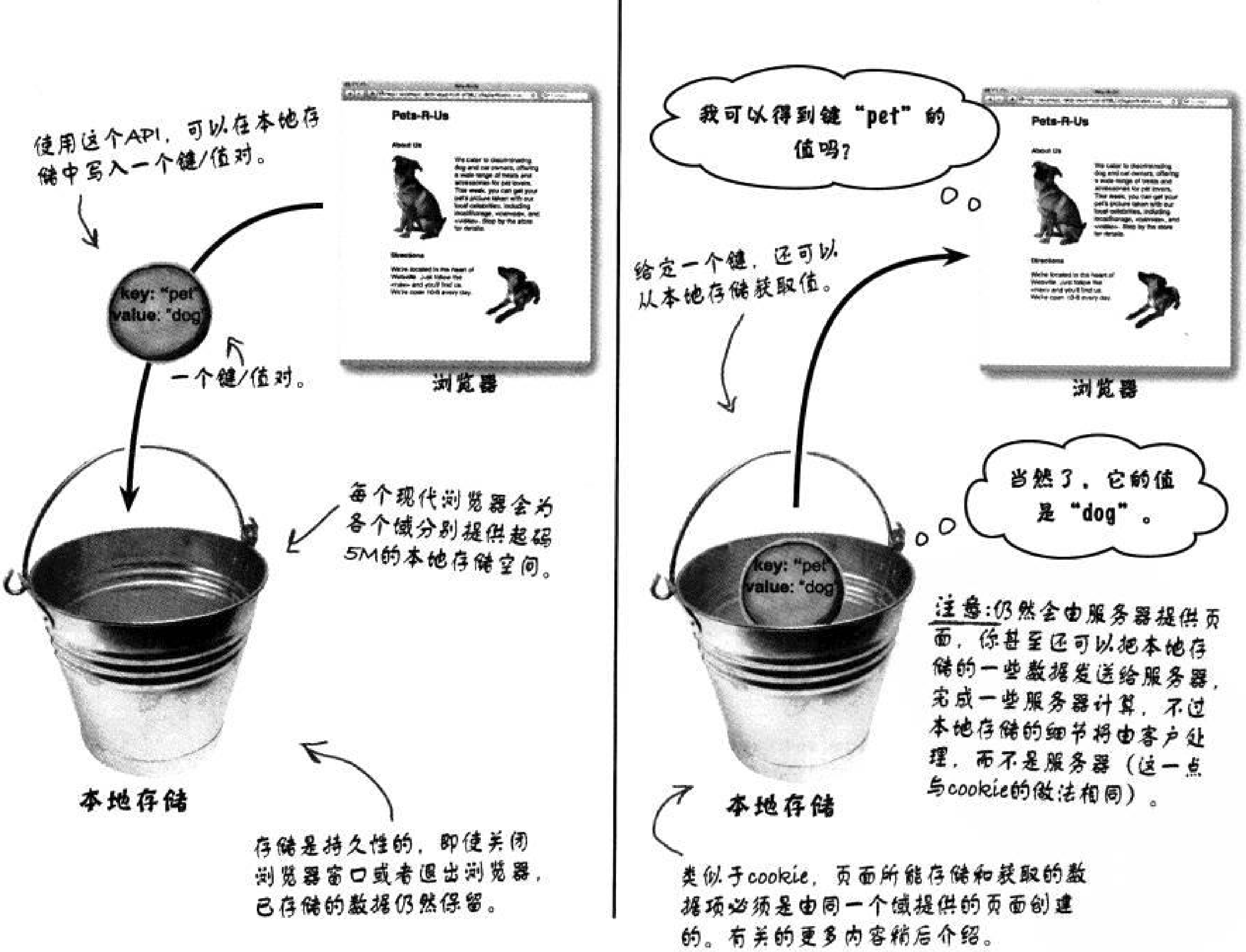
HTML5 Web存储如何工作

在幕后

HTML5确实为我们提供了一个很棒也很简单的JavaScript API，可以在浏览器中存储要持久存储的键/值对。现在不再仅限于少得可怜的4K存储空间，当今所有浏览器都很慷慨，每个用户的浏览器中都会提供5到10M的存储空间。创建HTML5的本地存储时还充分考虑了Web应用（和移动应用）。所谓本地存储，就是指你的应用可以把数据存储在浏览器上，从而减少与服务器之间所需的通信。下面来看这是如何做到的（然后我们会深入研究API）：

1. 页面可以在浏览器的本地存储中存储一个或多个键/值对。

2. 然后用键来获取相应的值。



记住……

需要一个系统来完成这些工作？要改进原先的即时贴系统（更常见的叫法是不干胶贴纸）可能很困难。你知道它是怎么做的：你把你"要做的事情"记下来，贴到某个地方，完成这个任务之后，就把即时贴扔到垃圾桶（或者回收）。

使用HTML来构建这样一个系统怎么样？下面来看看，我们需要一种方法来存储所有这些即时贴，所以需要一个服务器，还要有一些cookie……噢，等等，先别着急，我们完全可以用 HTML5 Web Storage API来做到！



高科技效能工具。

Web Storage API很简单，很有趣，而且你绝对会满意，真的，我们保证！



不再绕弯子了，下面就来使用本地存储。为此你要创建一个简单的HTML页面，包含所有基本部分：首部、体和一个脚本（或者可以直接使用代码示例中作为起步的文件notetoself.html）。把这些代码键入到<script>元素中（自己动手键入有助于你记住）：

1. 即时贴最重要的无非是你在上面写的文本，对不对？所以，首先存储一个写有"Pick up dry cleaning"的即时贴：

先从简单的做起，不过很快我们就会建立并运行一个完善的即时贴应用。

可以通过localStorage对象使用Web Storage API。你会发现浏览器已经为你定义了这个对象。使用这个对象时，就是在利用底层的本地存储系统。

setItem方法取两个字符串作为参数，它们将作为键/值对。

只能存储String类型的数据项。不能直接存储数字或对象（不过，我们很快就会想办法克服这个限制）。

```
localStorage.setItem("sticky _ 0", "Pick up dry cleaning");
```

要存储某个数据，可以使用setItem方法。

第一个字符串参数是键，数据项将按这个键存储。可以指定你希望的任何名字（只要它是一个字符串）。

第二个字符串是希望在本地存储中存储的值。

pin51.com 最新最全的资源下载站 拼吾爱

2 这太容易了：下面再向本地存储增加第二个数据项：

```
localStorage.setItem("sticky_1", "Cancel cable tv, who needs it now?");
```

另一个键，之前说过，可以使用你喜欢的任何键（只要它是一个字符串），但是每个键只能存储一个值。

对应这个新键的值。

3 既然在浏览器的本地存储中安全地存储了两个值，现在可以使用其中某个键从localStorage获取相应的值。如下：

我们将从本地存储得到与键"sticky_0"关联的值……

……并把它赋给名为sticky的变量。

```
var sticky = localStorage.getItem("sticky_0");

alert(sticky);
```

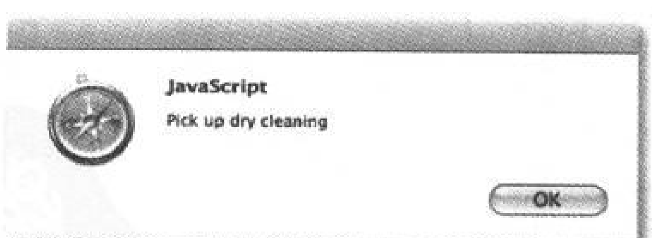
为了更有意思一些，下面使用alert函数在屏幕上弹出即时贴的值。

该试一试了！

确保将所有这些代码键入到script元素中，然后在浏览器中加载。

我们的测试结果如下：

这是我们的JavaScript提醒，sticky_0的值作为提醒消息。



有一点很棒：这个值是在浏览器的localStorage存储和获取的！你可以把浏览器关掉，放心地去富士山旅行一个月，等你回来时它还会在这里等着你。

没错，我们承认这个例子还可以更有趣一些，来吧，很快我们就会介绍一些更有意思的内容……

pin51.com 拼吾爱



当然，先在这里简单说一下：浏览器为你提供了一个本地存储，这是你自己的计算机上的空间，就在你的浏览器中，页面可以利用这个本地存储来存储键/值对。你创建了一些键/值对，使用本地存储API来存储这些键/值对，然后可以获取其中一个键/值对在应用中使用。尽管目前这个例子还不太让人兴奋，但要知道，利用每个用户浏览器中的一点点存储空间，你确实可以做很多有趣的事情（相信你至少能有一些想法）。

现在你大致有所了解了，下面就来详细地一步一步分析究竟发生了什么：

1 **首先，要记住每个浏览器有一些本地存储，可以用来存储键/值对。**



pin51.com 拼吾爱 最新最全资源分享下载站

2 有了这个本地存储，你可以取一个键和一个值（都采用字符串形式），把它们存储起来。

```
localStorage.setItem("sticky_0", "Pick up dry cleaning");
```

使用setItem方法来存储一个键/值对。键是“sticky_0”，值是“Pick up dry cleaning”。



调用setItem创建的键/值对。

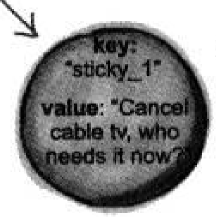


一旦把键/值对放在localStorage中，就会持久存储，即使你关闭了浏览器窗口，退出浏览器甚至重启计算机，它们仍然保留。

本地存储

3 然后再次调用setItem，再存储第二个键/值对，这一次键为“sticky_1”，值是“Cancel cable tv, who needs it now?”。

```
localStorage.setItem("sticky_1", "Cancel cable tv, who needs it now?");
```



现在有两个值，分别对应两个唯一的键存储。



本地存储

4 调用getItem并指定键“sticky_0”时，它会返回这个键/值对的值。

```
localStorage.getItem("sticky_0");
```

返回

“Pick up dry cleaning”

getItem查找一个键等于“sticky_0”的数据项（如果存在），并返回它的值。

注意，获取一个数据项时，并没有将它从本地存储删除，它还在那里。我们只是得到对应指定键的值。

pin51.com 拼吾爱

there are no Dumb Questions

问： **刚开始你说“Web存储”，后来你又开始说“本地存储”。它们一样吗？**

答： 这个Web标准正式的名字是“Web存储”（Web Storage），但是大多数人只是把它叫做本地存储（实际上，浏览器就是通过localStorage对象提供这个API的）。Web存储实际上并不是这个标准最合适的名字（因为数据项存储在你的浏览器中，而不是存储在Web上）。不过既然这么取名了，我们就只好沿用了。你会看到我们会更多地使用本地存储，而不是标准称呼“Web存储”。

问： **Web Storage API得到了广泛支持吗？能肯定总能得到它吗？**

答： 是的。实际上这是得到较好支持的API之一，甚至后退到IE8都能提供支持，而且现在大多数移动浏览器都支持这个API。可能有些浏览器会存在一些问题，谈到有关内容时我们会及时指出。至于能不能确信有Web Storage，与往常一样，在使用API之前还是应当先测试一下。可以如下测试localStorage：

```
if (window["localStorage"]) {
    // your localStorage code here……
}
```

注意，我们测试时会查看window全局对象是否有一个localStorage属性。如果有，则说明这个浏览器支持localStorage。

问： **这一章最前面你提到每个浏览器上有5MB的存储空间。这5MB是所有应用共用的吗？**

答： 不，实际上是每个域有5MB。

问： **你说过，不需要涉及服务器，可是后来你又开始提到域。**

答： 没错，所有存储空间都在客户端管理。引入域是因为5MB会分配给来自同一个域的所有页面作为存储空间。Pet-R-Us.com得到5MB，PetEmporium.com得到5MB，等等，所有这些都在你的机器上。

问： **这与Google Gears[或者可以在这里插入你最喜欢的专用本地存储技术]相比怎么样呢？**

答： 其他浏览器存储技术并没有什么问题，不过HTML5的本地存储现在已经作为标准（Google、Apple、Microsoft和其他公司现在都认可Web存储作为在浏览器本地存储内容的标准）。

问： **如果我对同一个键多次完成setItem会发生什么。假设我对“sticky_1”调用了两次setItem，会怎么样呢？本地存储中会得到两个sticky_1值吗？**

答： 不会。localStorage中键是唯一的，所以setItem会用第二个值覆盖第一个值。下面给出一个例子：如果运行这个代码：

```
localStorage.setItem("sticky_1","Get Milk");
localStorage.setItem("sticky_1","Get Almond Milk");
var sticky = localStorage.getItem("sticky_1");
```

sticky的值将是“Get Almond Milk”。

问： **谁能看到我的本地存储中的数据？**

答： 本地存储按数据的来源管理（可以把来源认为是域）。所以，例如，wickedlysmart.com上的每个页面可以看到这个网站上其他页面存储的数据项，但是其他网站（如google.com）的代码就无法访问这个存储空间（它们只能访问自己的本地存储数据项）。

问： **如果从我的计算机加载一个页面，就像前面的练习那样，那么我的数据源是什么？**

答： 这个问题问得好。在这种情况下，你的数据源称为“本地文件”源，这很适合用来进行测试。如果你可以访问服务器，也可以在服务器测试你的文件，这种情况下数据源就是服务器的域。

Watch it!

如果使用file://，本地存储可能无法在所有浏览器上都能正常工作。

有些情况下，一些浏览器会要求使用localhost://或一个托管服务器提供页面，而不是从一个文件加载，这就属于这种情况。所以，如果你的即时贴系统不能正常工作，可以尝试从一个服务器运行，或者也可以尝试使用一个不同的浏览器。

pin51.com 拼吾爱 最新最全的下载资源分享站

这么说，我可以在localStorage中存储内容，不过如果我想存储一个数呢？我一直希望可以使用localStorage为我将要编写的一个购物车应用存储商品个数（整数）和价格（浮点数），难道这个技术用不上吗？



Joel

这个技术非常适用。

没错，利用localStorage，只能使用字符串作为键和值。但是，这并没有听上去那么严格。假设你需要存储整数5。可以存储字符串“5”，然后在从本地存储获取时再将它转换回一个整数。下面就来看如何处理整数和浮点数。

假设你希望对应键“numitems”存储一个整数，可以写作：

```
localStorage.setItem("numitems", 1);
```

什么？刚才不是说不能存储整数吗？

是的，看起来这里像是在存储一个整数，不过JavaScript知道这必须是一个字符串，所以它会帮你将整数值强制转换为一个字符串。setItem真正看到的是字符串“1”，而不是一个整数。不过用getItem获取一个值时，JavaScript没有这么聪明：

```
var numItems = localStorage.getItem("numitems");
```

在这个代码中，numItems赋值为字符串“1”，而不是我们希望的整数。为了确保numItems是一个数字，需要使用JavaScript函数parseInt，将字符串转换为一个整数：

将值包装在一个parseInt调用中，它会把字符串转换为一个整数。

```
var numItems = parseInt(localStorage.getItem("numitems"));
numItems = numItems + 1;
localStorage.setItem("numitems", numItems);
```

可以让它加1，因为这是一个数字。

然后再次存储，再次由JavaScript负责转换。

如果存储浮点数值，从localStorage得到价格数据时，你可能要使用parseFloat函数：

这里完成同样的处理，我们存储一个浮点值，它会强制转换为一个字符串。

```
localStorage.setItem("price", 9.99);
var price = parseFloat(localStorage.getItem("price"));
```

再用parseFloat将它转换回一个浮点数。

pin51.com 拼吾爱 最新最全电子书下载站

本地存储和数组是双胞胎吗？

本地存储还有你不曾看到的一面。localStorage不仅提供了获取方法和设置方法（也就是getItem和setItem），它还允许你把localStorage对象看作是一个关联数组。这是什么意思？是这样的，可以不使用setItem方法，而是像这样在本地存储中为键赋一个值：




```
localStorage["sticky_0"] = "Pick up dry cleaning";
```

这里的键看起来就像是存储数组的索引。

这里把值放在赋值语句的右边。

还可以采用这种方式获取对应一个键存储的值。语法如下：

```
var sticky = localStorage["sticky _ 0"];
```

这就像使用getItem方法调用。

这里将变量sticky赋为……

……本地存储中“sticky_0”的值。

不错吧？不论使用哪种语法，都是合法的。不过，如果你习惯使用JavaScript中的关联数组，这个语法可能更简洁，而且对你来说更可读。

不过先等等，还有呢！

localStorage　API还提供了另外两个有意思的特性：一个属性length，还有一个方法key。length属性包含本地存储中的数据项数。从下面的代码可以看出key方法的作用：

这里迭代处理各个数据项。

length属性告诉我们localStorage中有多少个数据项。

总结：我们使用length迭代处理localStorage的内容（就像一个数组），并在迭代处理过程中访问各个键（如“sticky_0”）。然后使用这个键抽取相应的值。

```
for (var i = 0; i < localStorage.length; i++) {
        var key = localStorage.key(i);
        var value = localStorage[key];
        alert(value);
}
```

key方法会给出localStorage中各个数据项的键（如“sticky_0”、“sticky_1”等）。

试一试……是不是对应每个数据项得到一个提醒？

然后利用这个键名可以获取值。

pin51.com 拼吾爱

there are no Dumb Questions

问： 使用localStorage.key迭代处理localStorage时，会按什么顺序处理这些数据项？与我在本地存储中写入数据项的顺序一样吗？

答： 实际上并没有明确定义数据项的顺序。这是什么意思？这说明通过迭代，你会看到本地存储中的每一个键/值，但是在代码中不能依赖某种特定的顺序。事实上，不同的浏览器对同样的代码和数据项会给出不同的顺序。



果壳游戏

想试试运气吗？或者是不是该说想试试你的水平吗？这里有一个游戏，可以测试一下你对localStorage的掌握程度，不过一定要专心。利用你目前掌握的在localStorage中获取和设置键/值对的知识，跟踪豌豆到底在哪个果壳下面。

可以利用以下空白来记录localStorage的状态。

```
function shellGame() {
    localStorage.setItem("shell1", "pea");
    localStorage.setItem("shell2", "empty");
    localStorage.setItem("shell3", "empty");
    localStorage["shell1"] = "empty";
    localStorage["shell2"] = "pea";
    localStorage["shell3"] = "empty";
    var value = localStorage.getItem("shell2");
    localStorage.setItem("shell1", value);
    value = localStorage.getItem("shell3");
    localStorage["shell2"] = value;
    var key = "shell2";
    localStorage[key] = "pea";
    key = "shell1";
    localStorage[key] = "empty";
    key = "shell3";
    localStorage[key] = "empty";

    for (var i = 0; i < localStorage.length; i++) {
        var key = localStorage.key(i);
        var value = localStorage.getItem(key);
        alert(key + ": " + value);
    }
}
```

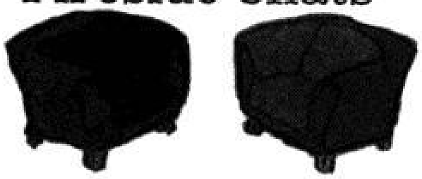
你也可以键入这个代码，检查你的答案是否正确，看看豌豆在哪个果壳下面。

哪个果壳下面有豌豆？把你的答案写在这里：

键	值
shell1	
shell2	
shell3	

pin51.com 拼吾爱

Fireside Chats



今晚话题：Cookie和本地存储

今晚我们请到了现任浏览器存储技术“Cookie”，以及目前新兴的领先技术“本地存储”。

Cookie：

就是你呀，现在的大红人，本地存储。我在这个行当已经闯荡了十多年了，你觉得你能像我一样吗？就好像你懂得不少一样。你还乳臭未干呢，小子，不是吗？

本地存储：

当然，你可以这么看，或者也可以这么讲，我就是根据从你的错误获得的所有那些经验建立的。

Cookie：

你知不知道多少个页面用到我？看过关于你的统计吗？

本地存储：

再过几年看看。事实上，我的目标是帮助建立浏览器的新一代Web应用。你提到的那些页面，大多都只是页面而已。

Cookie：

嘿，要知道我可是无处不在，哪里都有我的身影！我不会考虑是桌面浏览器、设备还是移动浏览器，不管浏览器有多老，你都能找到我。

本地存储：

我已经迎头赶上了。在所有HTML5技术中，我得到了最好的支持。

Cookie：

让我们拭目以待。那么你觉得你有哪些方面强过我呢？我的存储做得已经很好了。

本地存储：

呵呵，我真不想当着大家说这个，不过你确实存在大小限制的问题。

Cookie：

我不知道你在说些什么。

本地存储：

嘿，是你开的头。你很清楚你只限于4K的存储空间。而我是你的1200倍！

pin51.com 拼吾爱

Cookie：

是啊，我很轻巧，灵活，甚至可以说是敏捷。

没问题呀，我很开放的，你想放什么都可以在我这里存储。

嗯，键/值对算是重大的创新吗？

<窃笑>　呵，没错，你把所有东西都存储为字符串！干得真好！ </窃笑>

好吧，好吧，过十年再来找我吧，我们倒要看看你能不能经得过时间的考验。

等着瞧吧，等他们说，“哈哈，5MB，这就是你的优势吗？”那个时候你会哭着来找我的。

本地存储：

哈，说得真好听。你跟真正的Web开发人员聊过吗？你是有一些特点，但绝对不是敏捷。假设你要做统计，使用cookie能统计出开发人员因为愚蠢的错误和误解而损失的时间吗？

你实际上是说，你根本没有任何数据格式，所以开发人员必须发明一种新的机制在cookie中存储数据。

关于存储我们并不需要重大创新。键/值对工作得很好，很简单，而且适用于很多计算应用。

从字符串可以得到很多东西，如果你需要更复杂的，我还有很多其他办法。

噢，不信咱们可以打个赌。面对现实吧，从一开始你的命运就已经注定了。我是说，想想看，谁会把自己的孩子叫做Cookie？

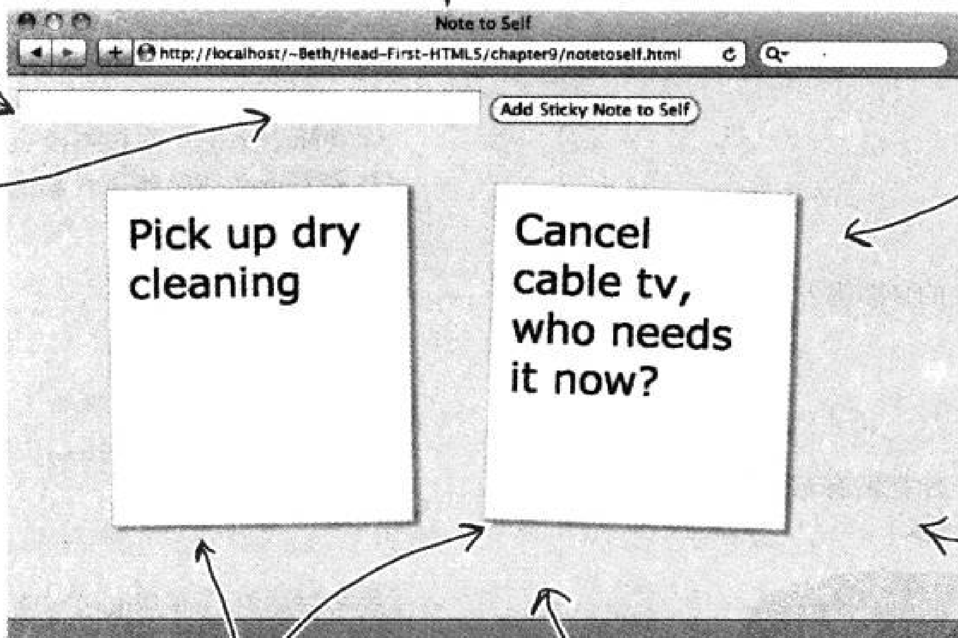
pin51.com 拼吾爱

处理即时贴

既然你有点时间来尝试Web存储，下面再进一步扩展这个实现。我们将创建一个即时贴应用，让你能看到你的即时贴，还可以增加新的即时贴。在构建这个应用之前，先来看看我们要构建的应用最终是什么样。



本地存储



pin51.com 拼吾爱

创建界面



首先，需要一种方法输入便条贴的文本。如果能在页面上看到就太好了，所以我们需要一个元素来包含页面上所有便条。

下面编写一些代码来完成这个工作，先从HTML标记开始，根据你现有的HTML文件，增加一个`<form>`元素、`<ul>`元素和CSS链接，如下所示：

这是我们的主HTML文件。

```
<!doctype html>
<html>
<head>
<title>Note to Self</title>
<meta charset="utf-8">
<link rel="stylesheet" href="notetoself.css">
<script src="notetoself.js"></script>
</head>
<body>
    <form>
        <input type="text" id="note_text">
        <input type="button" id="add_button" value="Add Sticky Note to Self">
    </form>

    <ul id="stickies">
    </ul>
</body>
</html>
```

我们将加入一些CSS，让它们看上去就像真正的即时贴。这本书并不是关于CSS的，不过如果需要你可以查看CSS源代码！

我们将把所有JavaScript移到文件“notetoself.js”中。

我们增加了一个表单作为用户界面，可以输入新的即时贴。

需要在界面的某个位置放置我们的即时贴，所以我们将把它们放在一个无序列表中。

利用CSS使各个列表项看上去就像真正的小贴纸。

pin51.com 拼吾爱 最新最全资源下载站

现在来增加JavaScript

我们已经有了页面所需的全部内容，而且localStorage中也已经有两个便条贴等着显示。下面就在页面上显示这两个即时贴，首先从localStorage将它们读出，然后放置在我们刚创建的无序列表元素中。可以这样做：

```
window.onload = init

function init() {
    for (var i = 0; i < localStorage.length; i++) {
        var key = localStorage.key(i);
        if (key.substring(0 , 6) == "sticky") {
            var value = localStorage.getItem(key)
            addStickyToDOM(value);
        }
    }
}
```

页面加载时，我们将调用init函数……

……它会从localStorage读取所有现有的即时贴，并通过DOM把它们增加到<ul>。

为此要迭代处理本地存储中的所有数据项。

获取各个键。

然后检查数据项的键是否以“sticky”开头，来确保它是一个即时贴。为什么要这么做？嗯，localStorage中可能还存储着其他数据项，而不仅仅是我们的即时贴（稍后会介绍更多有关内容）。

如果确实是一个即时贴，则获取它的值，并（通过DOM）增加到页面。

所以现在需要编写addStickyToDOM函数，向<ul>元素插入便条：

```
function addStickyToDOM(value) {
    var stickies = document.getElementById("stickies");
    var sticky = document.createElement("li");
    var span = document.createElement("span");
    span.setAttribute("class", "sticky");
    span.innerHTML = value;
    sticky.appendChild(span);
    stickies.appendChild(sticky);
}
```

传入便条贴的文本。我们需要为无序列表创建一个列表项，然后插入这个列表项。

所以，下面得到“stickies”列表元素。

创建一个列表元素，指定类名为“sticky”（以便指定样式）。

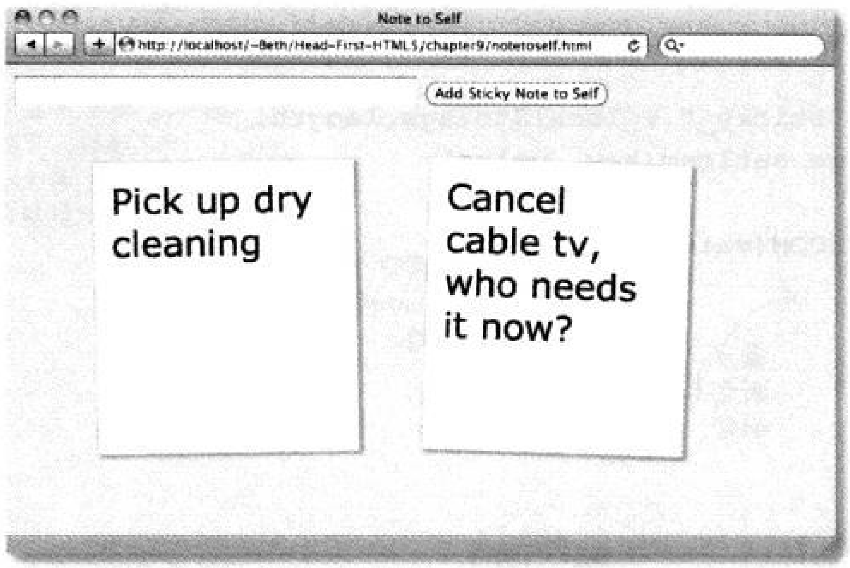
设置包含即时贴文本的span的内容。

将这个span增加到“sticky” li，再把这个li增加到“stickies”列表。

再来做个测试！

将这个代码键入到你的script元素中，并在浏览器中加载。

在浏览器中加载页面时会得到：



完成用户界面

现在要做的就是让这个表单起作用，能够增加新的便条。为此，需要为“Add Sticky Note to Self”按钮点击事件增加一个处理程序，并编写一些代码来创建一个新的即时贴。以下是增加处理程序的代码：

把这个代码增加到init函数。

```
function init() {
    var button = document.getElementById("add_button");
    button.onclick = createSticky;

    // for loop goes here
}
```

下面获取“Add Sticky Note to Self”按钮的引用。

并为这个按钮点击事件增加一个处理程序。将这个处理程序命名为createSticky。

init中的其余代码保持不变，为了节约纸张（相应地少砍树），这里不再重复。

pin5i.com 最新最全 拼吾爱 免费下载站

下面增加代码来创建一个便条贴：

点击按钮时，会调用这个处理程序。

它首先获取表单文本框中的文本。

```
function createSticky() {
    var value = document.getElementById("note_text").
value;
    var key = "sticky_" + localStorage.length;
    localStorage.setItem(key, value);

    addStickyToDOM(value);
}
```

然后需要为这个即时贴创建一个唯一的键。下面使用"sticky_"作为前缀，并连接整个本地存储的长度，从而构成键名，它会不断递增，对不对？

然后使用这个键向localStorage增加一个新的即时贴。

最后，向DOM增加新文本来表示这个即时贴。

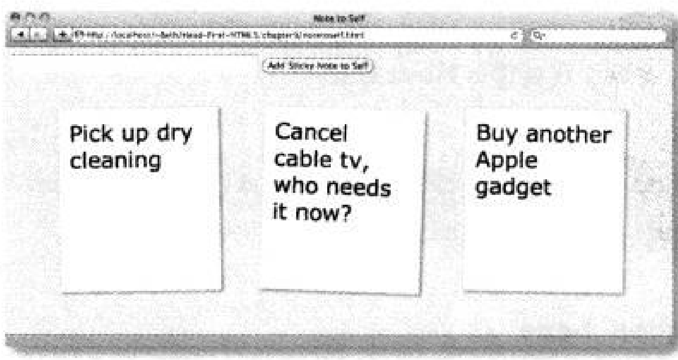
再做一个测试！



现在已经真正做到了交互！在浏览器中加载这个新代码，输入一个新的"便条贴"，点击或轻击"Add Sticky Note to Self"按钮。你会看到这个新的便条贴出现在你的即时贴列表中。我们会看到这样的结果：

你现在可以去富士山旅行了，等你回来，你的即时贴还在这里等着你！

这个即时贴的键是"sticky_2"，这是本地存储的长度（增加这个即时贴之前）与"sticky_"连接构成的。



这是测试运行的结果，看起来不错！

试着关闭浏览器，再重新打开这个文件。你还会看到这些即时贴！

pin51.com 拼吾爱

there are no Dumb Questions

问：为什么我们要测试查看各个数据项的键是不是以串“sticky”开头？

答： 应该记得，一个域（如apple.com）的所有页面会看到这个域中其他页面存储的各个数据项。这说明，如果我们对键的命名不在意，可能会与另一个采用不同方式使用相同键的页面发生冲突。所以，我们利用这种方法进行检查，在用它的值增加即时贴之前，先确保一个数据项确实是即时贴（而不是一个顺序号或者一个游戏等级）。

问：如果localStorage中有很多数据项，包括大量不是即时贴的数据怎么办？迭代处理整个数据项集合是不是太低效了？

答： 嗯，除非你所说的确实是非常非常多的数据项，否则我们很怀疑你会注意到效率的差别。不过，你说的也没有错，这样确实不太高效，可能会有更好的方法来管理键（稍后我们就会讨论这样一些方法）。

问：我不明白为什么使用localStorage.length作为键中的即时贴序号：

```
"sticky_" + localStorage.length
```

为什么要这么做？

答： 我们需要一种方法来创建唯一的新键。可以使用时间，或者生成一个每次都会递增的整数。或者，正如我们所说的，可以使用本地存储的长度（每次增加一个数据项时，长度都会递增）。如果你认为这样有问题，后面我们还会再做解释。如果你还没有考虑这会不会有问题，没关系，总之我们还会再来讨论的。

问：我在Safari中创建了一大堆的即时贴，然后切换到Chrome，在Chrome中看不到我的即时贴了，为什么呢？

答： 每个浏览器会维护它自己的本地存储。所以如果你在Safari中创建了即时贴，就只能在Safari中看到它们。

问：我刚刚重新加载了我的页面，现在即时贴顺序居然不一样了！

答： 增加一个新的便条贴时，我们的做法是把它追加到便条列表的最后，所以它总是位于列表的末尾。重新加载页面时，这些便条会按在localStorage中找到的顺序来增加（要记住，不能保证采用某种特定顺序）。你可能认为这个顺序就是数据项增加到本地存储时采用的顺序，或者另外某种合理的顺序，不过，不能指望这一点。为什么呢？一个原因是规范并没有指定一个顺序，所以不同的浏览器可能会以不同的方式实现这个顺序。如果你的浏览器确实按你可以理解的顺序返回数据项，那你很幸运，不过不能依赖于这个顺序，因为你的用户的浏览器可能会按不同的方式确定数据项的顺序。

问：我经常使用“for in”形式的for循环。这里能用吗？

答： 当然可以。可以这样使用

```
for (var key in localStorage) {
    var value = localStorage[key];
}
```

这会迭代处理localStorage中的各个键。很方便。

问：如果我不想要某个即时贴了怎么办？能不能删除？

答： 当然，我们可以从localStorage使用localStorage.removeItem方法删除数据项。还可以从localStorage直接使用浏览器控制台删除数据项。这两种做法在这一章都会介绍。

BRAIN POWER

根据实现这些即时贴的方式，如果用户能随便删除一个即时贴，我们的命名机制就可能存在问题。你能想出这个问题是什么吗？

pin51.com 拼吾爱 最新最全 下载站

需要暂停一下，来点预定服务

如果有一个工具可以直接查看localStorage中的数据项，是不是很好？或者，如果有一个工具能够删除数据项，甚至在你调试时能完全清除本地存储来从头开始，那该多好。

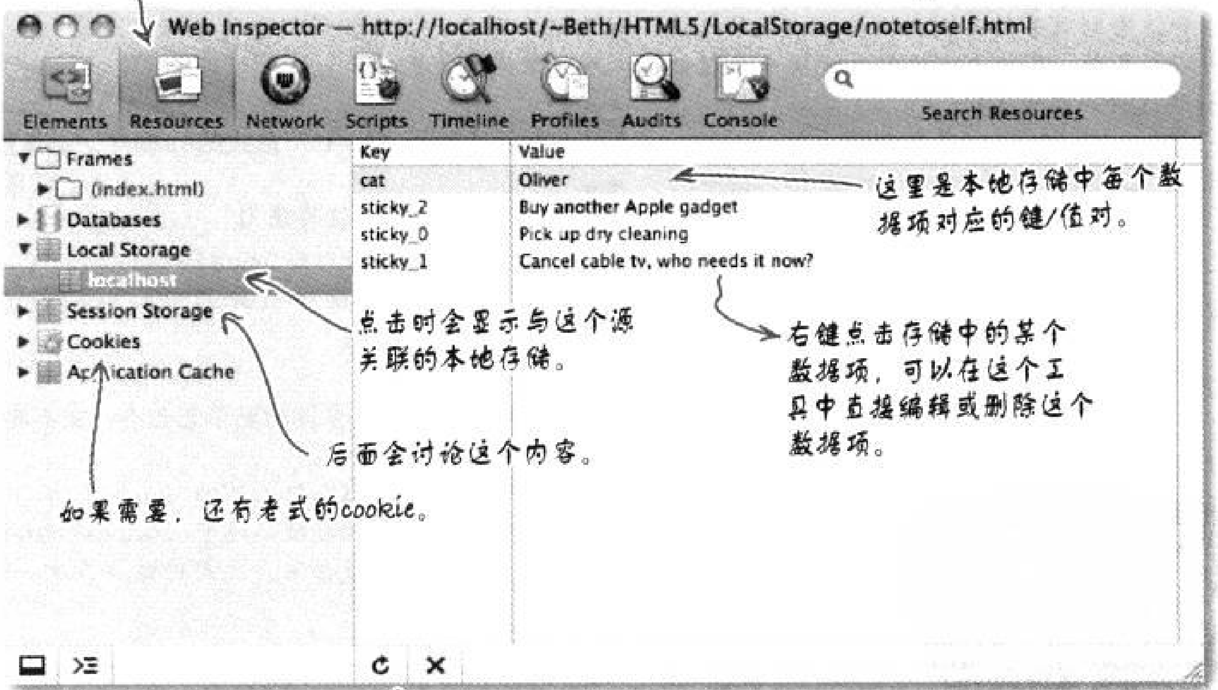


嗯，所有主流浏览器都内置有一些开发工具，允许直接检查你的本地存储。可以想见，不同浏览器中的工具各不相同，所以这里并不打算逐一地全面介绍，我们只会为你指出正确的方向，以便你进一步深入，明确你自己的浏览器的特定细节。不过，作为一个例子，下面来看Safari提供的工具：

更何况新版本的浏览器还在不断涌现，比我们写页面的速度还要快！

我们已经点击了Resources标签页，来查看localStorage。

Safari浏览器提供的开发工具。



这里是本地存储中每个数据项对应的键/值对。

点击时会显示与这个源关联的本地存储。

右键点击存储中的某个数据项，可以在这个工具中直接编辑或删除这个数据项。

后面会讨论这个内容。

如果需要，还有老式的cookie。

本地存储的源。这里我们使用由http://localhost提供的本地文件，不过，如果你在一个托管服务器上测试，这也可以是一个域名。

在Safari中，可以使用这些工具重新加载Storage视图，还可以删除所选择的数据项。

要启动或访问开发工具，正如我们说过的，对于不同的浏览器你要做的可能有所不同。在浏览器中加载`http://wickedlysmart.com/hfhtml5/devtools.html`，看看在你的特定浏览器上具体采用什么做法。

pin5i.com 拼吾爱 最新下载站

自己动手DIY维护

还有一种方法可以清除数据（稍后会看到，也可以逐个地删除），这需要你自己利用JavaScript完成一些维护工作。localStorage API包括一个很方便的方法clear，它会从你的本地存储删除所有数据项（至少，会删除你的域中的数据项）。下面就来看看如何在JavaScript中使用这个调用。我们将创建一个新文件，名为`maintenance.html`。创建这个文件之后，增加下面的代码，我们会逐步分析它是如何工作的。



这是一个很好的工具，可以放在你的工具箱里。

```
<!doctype html>
<html>
<head>
<title>Maintenance</title>
<meta charset="utf-8">
<script>
window.onload = function() {
    var clearButton = document.getElementById("clear_button");
    clearButton.onclick = clearStorage;
}

function clearStorage() {
    localStorage.clear();
}
</script>
</head>
<body>
    <form>
        <input type="button" id="clear_button" value="Clear storage" />
    </form>
</body>
</html>
```

我们已经向页面增加了一个按钮，这个代码会为这个按钮增加一个点击处理程序。

点击按钮时，会调用clearStorage函数。

这个函数所做的就是调用localStorage.clear方法。使用时要当心，因为它会删除与这个维护页面的源关联的所有数据项！

这里是我们的按钮。如果需要删除localStorage中的所有内容，都可以使用这个文件（这对测试很有好处）。

输入这个代码之后，在浏览器中加载（对于我们的即时贴应用）。现在可以安全地清空你的localStorage。可以试一试！一定要确保已经了解开发工具，这样才能观察到变化。

Watch it!

这会删除域中的所有数据项！

如果你有一个超级有价值的本地存储与同一个域中的另一个项目关联，运行这个代码后，你会丢失所有数据项。好好想想吧……

pin51.com 拼吾爱

> 我遇到一个问题。在做这本书中的练习时，同时还在利用我掌握的知识创建我们公司的新购物车应用。我的即时贴应用不能正常工作了。我用Safari开发工具查看localStorage时，发现我的即时贴编号一团糟，现在有“sticky_0”、“sticky_1”、“sticky_4”、“sticky_8”、“sticky_15”、“sticky_16”、“sticky_23”、“sticky_42”。我有一种感觉，发生这种情况是因为我在创建即时贴的同时还在localStorage中创建了其他数据项。这到底是怎么回事？

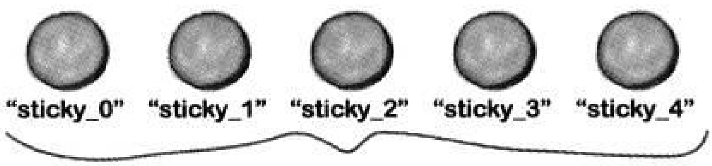


哈，你发现了一个重要的设计缺陷。

没错，是时候澄清这个问题了：到目前为止，我们已经建立了一个很不错的小应用，但前提是没有向localStorage引入其他数据项（比如Joel的购物车）。一旦引入了其他数据项，我们用来跟踪即时贴的整个机制就不再奏效，或者，至少不能很好地工作了。下面来告诉你原因：

如果你能接受这个要求，那很好；否则，最好继续读下面的内容。

首先，我们的便条贴是按从0到即时贴数目（减1）来编号的：

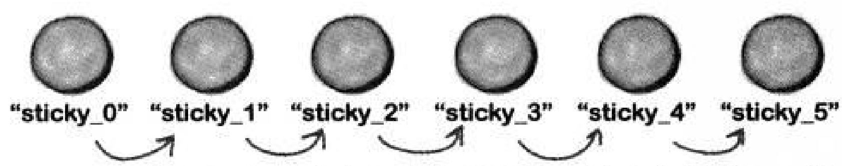


要增加一个新的即时贴，我们会统计本地存储中的数据项个数，并从这个数开始创建新的键：

```
var key = "sticky_" + localStorage.length;
```

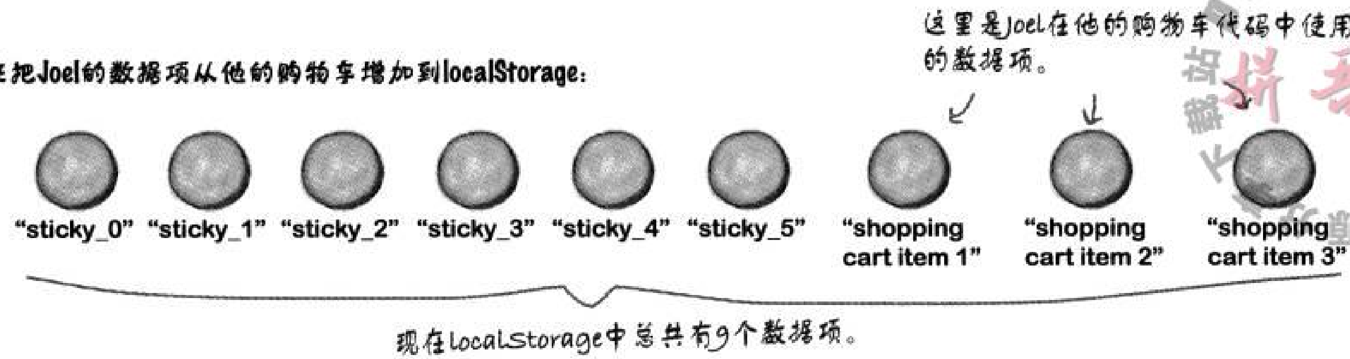


要显示所有即时贴，则从0开始，迭代处理本地存储中的所有数据项，直到本地存储的长度（减1）：



现在长度为6，所以从0迭代处理到5，显示从“sticky_0”到“sticky_5”的各个便条。

现在把Joel的数据项从他的购物车增加到localStorage：



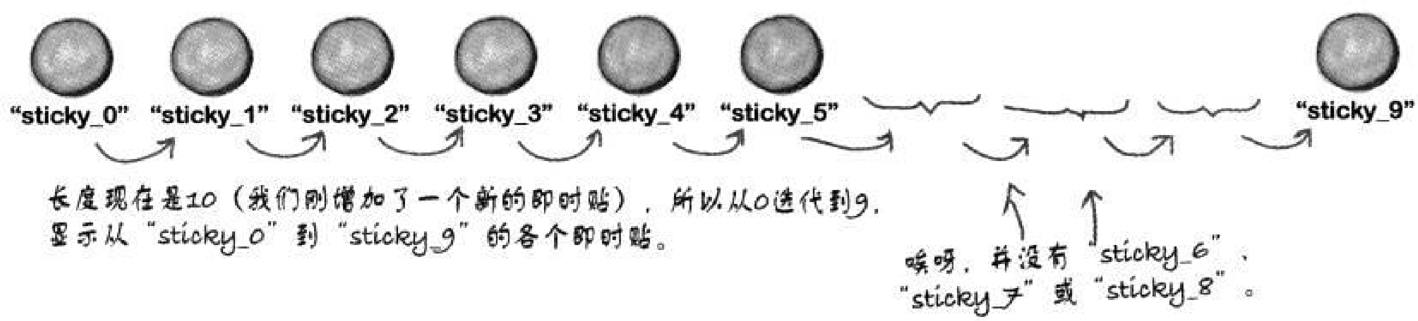
下面创建一个新的即时贴：

```
var key = "sticky_" + localStorage.length;
```



创建新的即时贴时，本地存储的长度现在是9，所以我们会创建一个名为"sticky_9"的便条。嗯，看起来不太对劲。

需要迭代处理即时贴来显示时，我们就遇到麻烦了：



Sharpen your pencil

你认为我们当前的实现在哪些方面可能会有问题，请在旁边打勾：

- ☐ 如果localStorage中有很多数据项不是即时贴，显示即时贴会很低效。
- ☐ 如果另一个应用删除它自己的数据项而导致localStorage长度变小时，即时贴可能会被setItem覆盖。
- ☐ 很难很快说出到底有多少个即时贴；必须迭代处理localStorage中的每一个数据项来得到所有即时贴。
- ☐ 用cookie吧，肯定比解决这些问题更容易！

pin51.com 拼吾爱 最新最全资源分享下载站



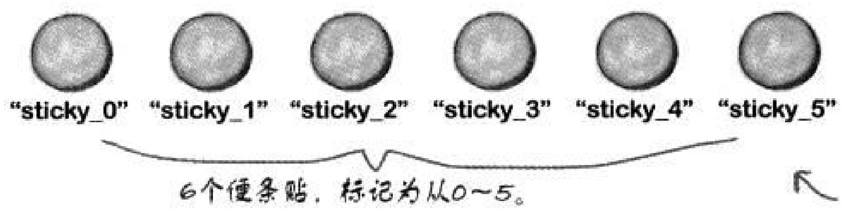
pin51.com 拼吾爱 最新最全资源分享下载站

这个技术确实有……

我们没有撒谎，只能存储字符串作为localStorage数据项的值，这一点确实是真的，不过这并不是全部，因为在具体存储之前，完全可以把一个数组（或一个对象）转换为字符串。当然，看起来好像是投机取巧，不过这确实是合法的做法，这样就可以在localStorage中存储非字符串数据类型了。

我们知道，你希望现在就深入了解存储数组的细节，不过在此之前，先来一步一步研究如何利用数组解决我们（和Joel）的问题。

先后退一步，假设localStorage中有6个即时贴：



即时贴和即时贴数组都存储在localStorage中。

localStorage中有一个数组，名为"stickiesArray"：

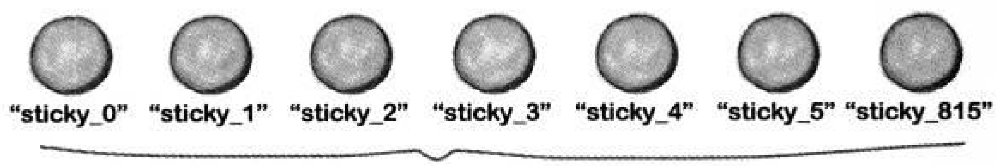
"sticky_0"	"sticky_1"	"sticky_2"	"sticky_3"	"sticky_4"	"sticky_5"

"stickiesArray"

即时贴数组中的每个元素分别是localStorage中一个即时贴的键。

现在来增加一个新的即时贴。称这个即时贴为"sticky_815"。为什么会有这么怪异的一个编号呢？因为我们不再关心它叫什么了，只要它是唯一的就行。所以，要增加这个即时贴，我们可以向数组增加"sticky_815"，然后为这个即时贴存储一个数据项，就像之前的做法一样。如下所示：

localStorage中多了一个即时贴。



"sticky_0"	"sticky_1"	"sticky_2"	"sticky_3"	"sticky_4"	"sticky_5"	"sticky_815"

"stickiesArray"

这个即时贴数组扩展了一个值。

pin51.com 拼吾爱 最新最全资源分享下载站

使用数组重新实现应用

OK，我们大致知道了如何使用数组来记录我们的即时贴，不过下面再更进一步，确保能迭代处理和显示所有即时贴。在当前的代码中，我们会在init函数中显示所有即时贴。能不能重写这个函数来使用一个数组呢？先来看现在的代码，然后再看如何用数组做出修改（希望同时能改进代码）。先不要键入这个代码，我们只强调现在需要做的一些修改，但不能保证这个代码安全到万无一失。稍后会介绍保证安全的内容。

之前的代码……

```
function init() {
    // button code here...
    for (var i = 0; i < localStorage.length; i++) {
        var key = localStorage.key(i);
        if (key.substr(0, 6) == "sticky") {
            var value = localStorage.getItem(key);
            addStickyToDOM(value);
        }
    }
}
```

这是我们原来的代码，依赖于即时贴特定的名字，sticky_0、sticky_1等……

哇，这太乱了，想想看吧。

现在我们知道了，这可能会出问题，因为如果根据localStrorage中数据项的个数来命名，并不能确定存在相应的即时贴。

改进后的新代码

```
function init() {
    // button code here...
    var stickiesArray = localStorage["stickiesArray"];
    if (!stickiesArray) {
        stickiesArray = [];
        localStorage.setItem("stickiesArray", stickiesArray);
    }

    for (var i = 0; i < stickiesArray.length; i++) {
        var key = stickiesArray[i];
        var value = localStorage[key];
        addStickyToDOM(value);
    }
}
```

首先从localStorage获取stickiesArray。

需要确保localStorage中有一个数组。如果没有，就创建一个空数组。

迭代处理这个数组。

数组中的各个元素分别是一个即时贴的键，所以我们要使用这个键从localStorage获取相应的数据项。

然后把这个值增加到DOM，就像前面做的一样。

说明：你还不知道如何在localStorage中存储和获取数组，所以在我们给出具体代码之前可以把它看作是伪代码。必须对这个伪代码做一个很小的补充，它才能正常工作。

pin51.com 拼吾爱



还需要确定具体如何在localStorage中存储一个数组。

你可能已经猜到，我们可以使用JSON来创建数组的一个字符串表示，如果是这样，那你猜对了。一旦想到这一点，就可以在localStorage中存储数组了。

应该记得，JSON API中只有两个方法：stringify和parse。下面具体运用这两个方法来完成init函数（在学习后面的内容之前，先对照检查本章最后给出的答案）：

```
function init() {
    // button code here...
    var stickiesArray = localStorage["stickiesArray"];
    if (!stickiesArray) {
        stickiesArray = [];
        localStorage.setItem("stickiesArray", ______________(stickiesArray));
    } else {
        stickiesArray = __________ (stickiesArray);
    }
    for (var i = 0; i < stickiesArray.length; i++) {
        var key = stickiesArray[i];
        var value = localStorage[key];
        addStickyToDOM(value);
    }
}
```

我们增加了这个else子句，因为如果从localStorage得到数组，你需要做一些工作（这是一个字符串而不是一个数组）。

转换createSticky来使用数组

这个应用就快要完成了。现在要做的就是修改createSticky方法，你应该记得，这个方法只是从表单得到即时贴的文本，在本地存储，然后显示出来，在修改这个函数之前先来看当前的实现：

```
function createSticky() {
    var value = document.getElementById("note_text").value;
    var key = "sticky_" + localStorage.length;
    localStorage.setItem(key, value);

    addStickyToDOM(value);
}
```

不再使用localStorage长度来创建键，之前我们已经看到这会带来问题，现在我们需要创建一个更为唯一的键。

还需要将这个即时贴增加到即时贴数组中，并把数组保存在localStorage中。

pin51.com 最新 下载站 拼吾爱

哪些需要修改？

createSticky中有两点需要修改。首先，我们需要一种新方法为每个即时贴生成一个唯一的键。另外还需要修改代码，把即时贴存储在localStorage中的stickiesArray数组中。

1 需要为即时贴创建一个唯一的键

有很多方法来创建唯一的键。我们可以使用日期和时间，也可以创建复杂的的随机64位数，或者在应用中使用一个原子时钟API。嗯，听上去日期和时间就不错，是达到这个目的的一种便捷方法。JavaScript支持一个Date对象，它能返回自1970年以来的毫秒数，这应该足够唯一了（除非你要以非常快的速率创建即时贴）：

创建一个Date对象，然后得到当前时间（毫秒数）。

```
var currentDate = new Date();
var time = currentDate.getTime();
var key = "sticky_" + time;
```

这是创建一个唯一键的新代码。

然后将这个毫秒数追加到字符串"sticky_"后面来构造键。

2 需要把这个新即时贴存储在数组中

既然已经有办法创建一个唯一的键，还需要存储对应这个键的即时贴文本，并把这个键增加到stickiesArray。下面介绍如何做到，然后再把所有这些代码集成在一起。

首先获取即时贴数组。

```
var stickiesArray = getStickiesArray();
localStorage.setItem(key, value);
stickiesArray.push(key);
localStorage.setItem("stickiesArray",
                     JSON.stringify(stickiesArray));
```

与（上一页）init函数中的做法不同，这里不再重复列出得到和检查stickiesArray的代码，我们将创建一个新函数完成这个工作。稍后就会介绍这个函数。

然后像往常一样，存储键和相应的值（不过要使用我们的新键）。

然后使用数组方法push，它会把这个键追加到即时贴数组的末尾。

再将这个数组存储到localStorage中，先将它转换为一个字符串。

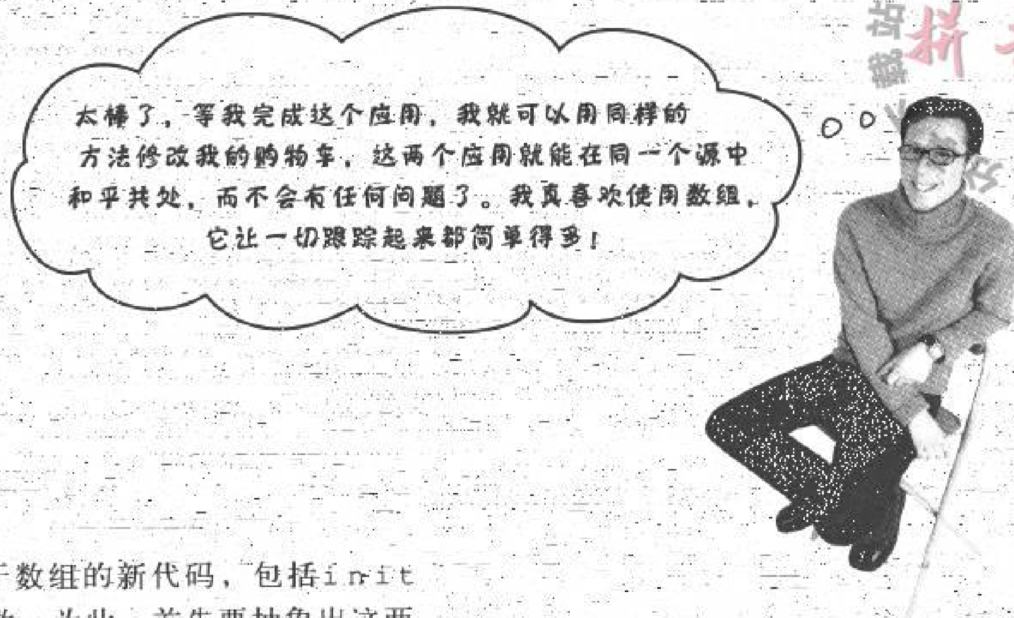
there are no Dumb Questions

问： 怎么是自1970年以来的毫秒数？

答： 你可能已经知道，毫秒就是千分之一秒，getTime方法会返回从1970年以来的总毫秒数。为什么是1970年？这是由Unix操作系统继承来的，它就是这样定义时间的。尽管这并不完美（例如，1970年之前的时间会用负数表示），但如果需要一个唯一的数或者要在JavaScript代码中跟踪时间，这确实很方便。

问： JSON类型的这些解析和串化工作是不是不够高效？如果我的数组非常大，那么存储时是不是效率也很低？

答： 理论上讲，确实是这样。但是，对于一般的Web页面编程任务，这往往不会成为问题。不过，如果你在实现一个正式的应用，有很大的存储需求，可能会发现使用JSON将数据项与字符串来回转换时会有问题。



集成在一起

现在来集成所有这些基于数组的新代码，包括`init`和`createSticky`函数。为此，首先要抽象出这两个函数中都需要的一小部分代码，这部分代码用来从`localStorage`获取即时贴数组。你在`init`中已经见过这些代码，在`createSticky`中还要用到它。下面就取出这部分代码，放它放在一个名为`getStickiesArray`的方法中，之前我们已经详细分析过这些代码，所以这个内容对你来说应该不陌生：

```
function getStickiesArray() {

    var stickiesArray = localStorage.getItem("stickiesArray");

    if (!stickiesArray) {

        stickiesArray = [];

        localStorage.setItem("stickiesArray", JSON.stringify(stickiesArray));

    } else {

        stickiesArray = JSON.parse(stickiesArray);

    }

    return stickiesArray;

}
```

首先从localStorage得到数据项“stickiesArray”。

如果这是第一次加载这个应用，可能还没有一个“stickiesArray”数据项。

如果还没有数组，就要创建一个空数组，然后再把它存入localStorage。

别忘了先把它转换为字符串！

否则，在localStorage中找到这个数组，需要进行解析，将它转换为一个JavaScript数组。

不论哪一种情况，最后都会得到一个数组，所以返回这个数组。

拼吾爱 最新资源 www.pin51.com

pin51.com 拼吾爱 最新最全的免费下载站

集成在一起（续）……

写好了getStickiesArray，下面来看简化后的最终版本的init和createSticky函数。输入下面的代码：

```
function init() {
    var button = document.getElementById("add_button");
    button.onclick = createSticky;

    var stickiesArray = getStickiesArray();

    for (var i = 0; i < stickiesArray.length; i++) {
        var key = stickiesArray[i];
        var value = localStorage[key];
        addStickyToDOM(value);
    }
}
```

记住，我们还要在init方法中建立按钮事件。

接下来获取包含即时贴键的数组。

现在要迭代处理这个即时贴数组（而不是localStorage数据项）。

数组中的每一项分别是一个即时贴的键。下面来获取各个键。

并从localStorage获取相应的值。

把它增加到DOM，就像前面所做的一样。

完成了init之后，只剩下createSticky了：

```
function createSticky() {
    var stickiesArray = getStickiesArray();
    var currentDate = new Date();
    var key = "sticky_" + currentDate.getTime();
    var value = document.getElementById("note_text").value;
    localStorage.setItem(key, value);
    stickiesArray.push(key);
    localStorage.setItem("stickiesArray", JSON.stringify(stickiesArray));
    addStickyToDOM(value);
}
```

首先获取即时贴数组。

然后为新即时贴创建唯一的键。

将即时贴键/值对增加到localStorage。

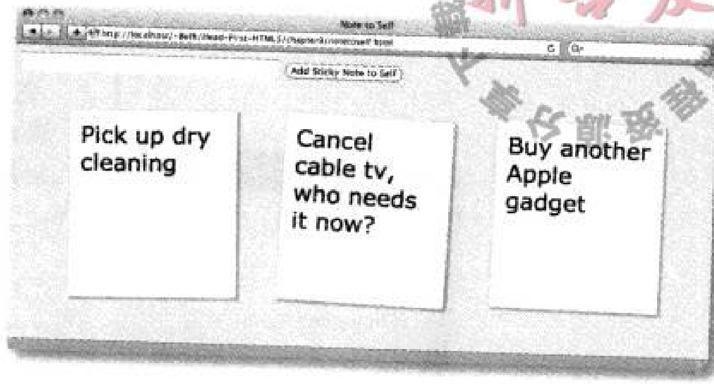
并把这个新键增加到即时贴数组……

然后将这个数组转换为字符串，写回到localStorage。

最后，用新即时贴更新页面，把这个即时贴增加到DOM。

试一试！

输入所有代码，并清空localStorage，清清爽爽地开始。加载这个代码，你应该能看到与前一次完全相同的表现。Joel，你会看到你的代码现在能正确地工作了！



there are no Dumb Questions

问： 我们一直在使用"sticky_"作为localStorage数据项名的前缀。对于localStorage命名机制有没有什么约定？

答： localStorage数据项的命名并没有约定。如果你的Web应用在域中的一个小网站上，而且你有能力控制这个域，那么命名就不会成为问题，因为你很清楚这个网站上所有不同页面使用的所有名字。最好使用一个特定的名字，可以指示依赖于这个数据项的页面或Web应用，我们认为这种想法很好。所以"sticky_"能帮助我们记住这些数据项都与即时贴应用有关。

问： 这么说，如果我的即时贴应用只是域中的多个应用之一，我就必须考虑潜在的冲突，是吗？

答： 没错。在这种情况下，对你来说（或者对于管理这个域中各网站的某个人），最好为如何对数据项命名做出规划。

问： 如果我有很多的即时贴，我的stickiesArray会变得很长。这会有问题吗？

答： 除非你创建了成千上万的即时贴，否则不会有问题（如果你确实创建了成千上万的即时贴，我们倒是很想知道你怎么能如此多产）。现如今，JavaScript的速度已经相当快了。

问： 那么再明确一下，我们可以在localStorage中存储任何对象，只需要先用JSON把它转换为串就行了，是吗？

答： 完全正确。JSON串是JavaScript对象的简化版本，而且大多数简单JavaScript对象都可以使用JSON转换为一个字符串，并存储在localStorage中。这包括数组（你已经看到了），以及包含属性名和值的对象，稍后就会看到。

要为你的localStorage数据项选择一个命名机制，从而不会与同一个域中的其他应用发生冲突。

如果需要在localStorage中存储数组或对象，可以使用JSON.stringify来创建要存储的值，在获取之后可以使用JSON.parse来解析。

pin51.com 拼吾爱 最新最全资源分享下载站



当心这些吹毛求疵的家伙！

删除便条贴

她说得没错，如果不能删除即时贴，这个应用不会有太大成功。这一章我们已经提到过`localStorage.removeItem`方法，不过还没有具体讨论。`removeItem`方法取一个数据项的键，并从localStorage将这个数据项删除：

```
localStorage.removeItem(key);
```

这个方法会删除localStorage中有指定键的数据项。

removeItem有一个参数，这是要删除的数据项的键。

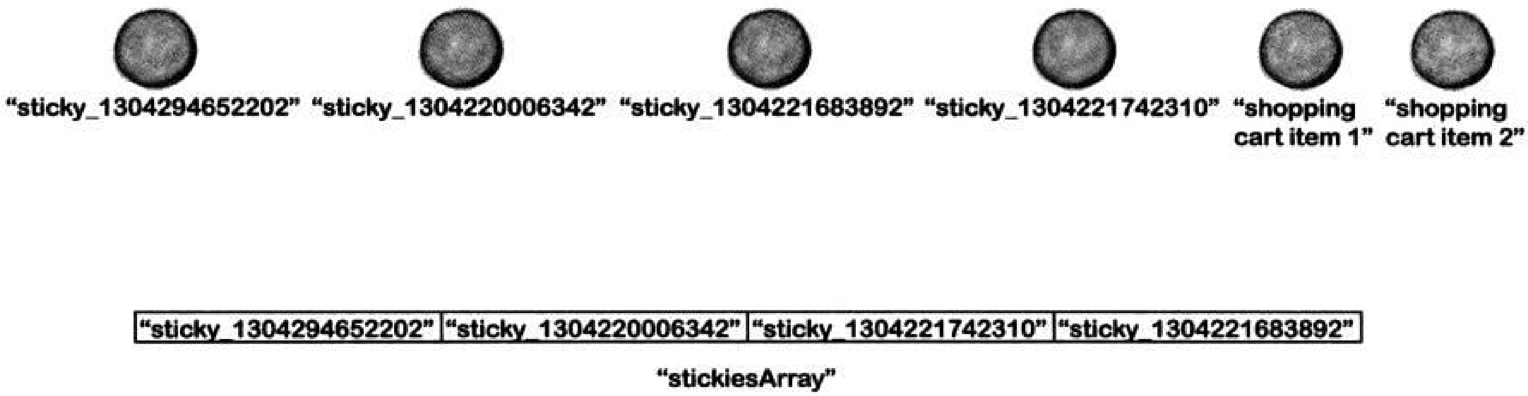
听上去很容易，是不是？哈，不过如果你再仔细想想，删除即时贴并不只是调用`removeItem`方法那么简单。我们还需要处理`stickiesArray`...

pin51.com 拼吾爱

Sharpen your pencil

删除一个即时贴!

下面你会看到localStorage的内容。你已经有了所需的全部JavaScript代码以及removeItem方法。拿出铅笔，画出从localStorage删除sticky _ 1304220006342时需要做什么。画完草图后，再在下面写出伪代码，说明你将如何编写代码。



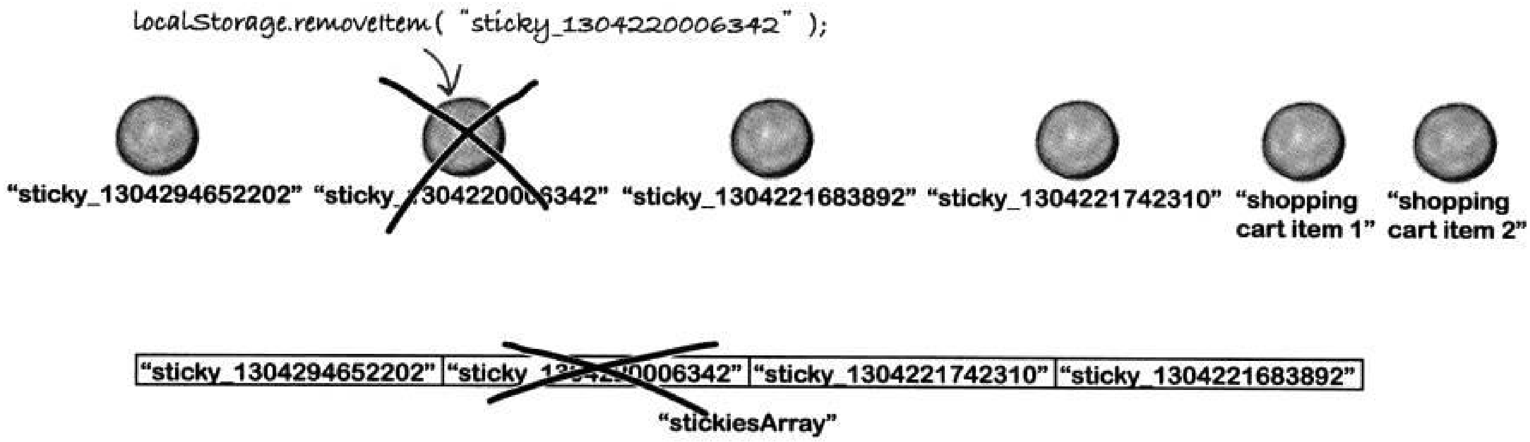
在这里写出你的伪代码。

pin51.com 拼吾爱

Sharpen your pencil Solution

删除一个即时贴！

下面你会看到localStorage的内容。你已经有了所需的全部JavaScript代码以及removeItem方法。拿出铅笔，画出从localStorage删除sticky_1304220006342时需要做什么。画完草图后，再在下面写出伪代码，说明你将如何编写代码。以下是我们的答案。



(1) 使用localStorage.removeItem方法从localStorage删除键为"sticky_1304220006342"的即时贴。

(2) 得到stickiesArray。

(3) 从stickiesArray删除key="sticky_1304220006342"的元素。

(4) 将stickiesArray写回到localStorage（先将它转换为一个字符串）。

(5) 在DOM中查找"sticky_1304220006342"，并将它删除。

pin51.com 拼吾爱 最新最全下载站 分享资源

deleteSticky函数

你已经对如何删除便条贴做出了计划，下面就来看一看deleteSticky函数：

```
function deleteSticky(key) {
    localStorage.removeItem(key);
    var stickiesArray = getStickiesArray();
    if (stickiesArray) {
        for (var i = 0; i < stickiesArray.length; i++) {
            if (key == stickiesArray[i]) {
                stickiesArray.splice(i,1);
            }
        }

        localStorage.setItem("stickiesArray", JSON.stringify(stickiesArray));
    }
}
```

首先，使用removeItem从localStorage删除即时贴，传入要删除的即时贴的键。

使用getStickiesArray函数从localStorage得到stickiesArray。

确保有一个stickiesArray（以防万一），然后迭代处理数组来查找我们想要删除的键。

找到这个键后，使用splice将它从数组删除。

splice会将数组中从第一个参数(i)指定的位置开始的元素删除，删除的元素个数由第二个参数(1)指定。

最后，将stickiesArray（已经将这个键删除）再保存到localStorage。

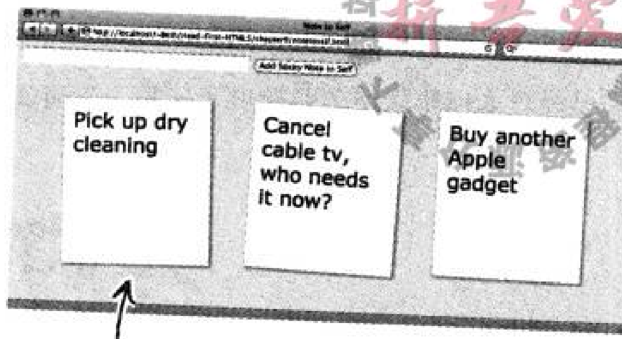


我得到了代码，但是没有看到如何得到传递给deleteSticky的键。想想看，刚开始用户如何选择要删除的便条呢？

pin51.com 拼吾爱

如何选择要删除的即时贴？

需要有一种方法允许用户选择要删除的即时贴。我们可以不遗余力地为每个便条增加一个小删除图标，不过，对于这个即时贴应用，我们想更简单一些：只是当用户点击某个即时贴时才将它删除。从可用性角度讲这可能不是最好的实现，不过，确实很简单。



点击一个便条贴时，会将它删除。

要实现这个功能，首先需要修改即时贴，以便检测出什么时候点击了这个即时贴，然后可以将这个即时贴传入deleteSticky函数。

addStickyToDOM函数中要做很多事情，下面来看如何完成：

总结：我们将使用即时贴的键来唯一标识便条，要记住，键为"sticky_"+时间。只要调用addStickyToDOM就会传入这个键。

```
function addStickyToDOM(key, value) {
    var stickies = document.getElementById("stickies");
    var sticky = document.createElement("li");
    sticky.setAttribute("id", key);
    var span = document.createElement("span");
    span.setAttribute("class", "sticky");
    span.innerHTML = stickyObj.value;
    sticky.appendChild(span);
    stickies.appendChild(sticky);
    sticky.onclick = deleteSticky;
}
```

为DOM中表示即时贴的<li>元素增加一个唯一的id。这样一来，deleteSticky就能知道你点击了哪个即时贴。由于我们已经知道即时贴的键是唯一的，所以只需使用这个键作为id。

我们还为各个即时贴增加了点击处理程序。点击一个即时贴时，就会调用deleteSticky。

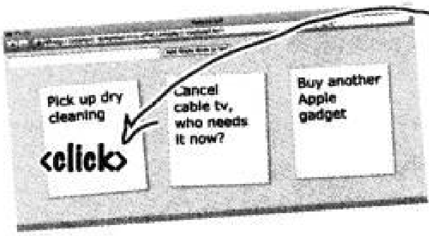
Exercise

现在你的任务是更新所有代码，只要调用addStickyToDOM，都要传入键和值。应该很容易找到需要更新的位置。不过，等你完成后，请对照检查本章最后给出的答案，看看你找得对不对。

不要跳过这个练习，否则后面的测试会不正常！

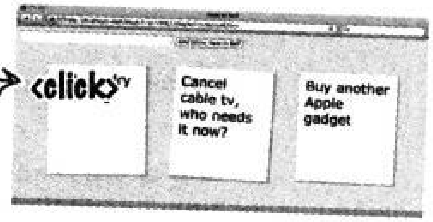
如何由事件得到要删除的即时贴

现在每个即时贴上有一个事件处理程序来监听点击事件。点击一个即时贴时，会调用deleteSticky，并把一个事件（event）对象传递到deleteSticky，同时还会传递有关这个事件的信息，如点击了哪个元素。我们可以查看event.target来得出点击了哪个即时贴。下面来仔细分析点击一个即时贴时到底发生了什么。



如果点击一个即时贴的黄色部分，事件目标将是<li>元素。这正是我们想要的，因为<li>现在的id就是这个即时贴的键。

如果点击文本，事件目标则是<li>内部的<span>，这可不是我们想要的。




```
<li id="sticky_1304270008375">
    <span class="sticky">Pick up dry cleaning</span>
</li>
```

这是addStickyToDOM中创建即时贴的HTML。

不论哪种情况，点击即时贴生成的事件都会传递到deleteSticky。

目标（target）就是所点击并生成事件的元素，可以从target属性得到这个元素的id。如果目标是<li>，就大功告成了。

```
function deleteSticky(e) {
    var key = e.target.id;
    if (e.target.tagName.toLowerCase() == "span") {
        key = e.target.parentNode.id;
    }
    localStorage.removeItem(key);
    var stickiesArray = getStickiesArray();
    if (stickiesArray) {
        for (var i = 0; i < stickiesArray.length; i++) {
            if (key == stickiesArray[i]) {
                stickiesArray.splice(i,1);
            }
        }
        localStorage.setItem("stickiesArray", JSON.stringify(stickiesArray));
        removeStickyFromDOM(key);
    }
}
```

如果目标是<span>，则需要得到它的父元素<li>的id。<li>元素的id才是我们需要的键。

现在可以使用这个键从localStorage以及stickiesArray删除数据项。

还需要从页面删除包含这个即时贴的<li>，这样一来，点击这个即时贴时它就会消失。接下来就做这个工作……

pin51.com 拼吾爱 最新资源分享下载站

还要从DOM删除即时贴

要完成删除工作，需要实现removeStickyFromDOM函数。之前已经更新了addStickyToDOM函数，在DOM中增加了即时贴的键作为包含即时贴的<li>元素的id，所以我们可以使用document.getElementById找到DOM中的即时贴。首先得到即时贴的父节点，并使用removeChild方法来删除即时贴：

传入所寻找的sticky元素的键（也就是id）

```
function removeStickyFromDOM(key) {
    var sticky = document.getElementById(key);
    sticky.parentNode.removeChild(sticky);
}
```

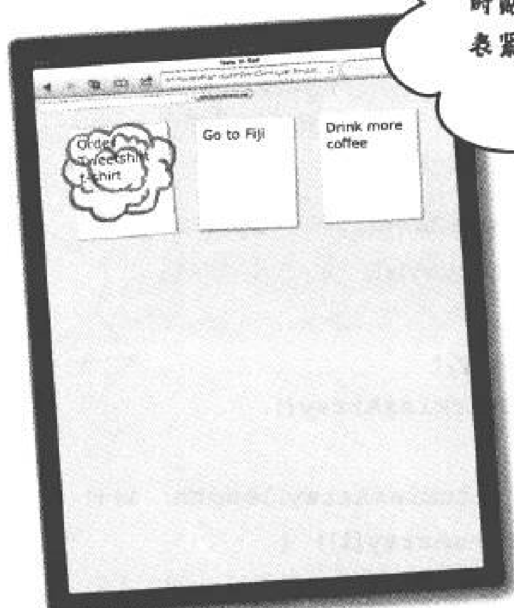
从DOM获取<li>元素……

……将其删除，首先得到其parentNode，然后使用removeChild将它删除。

  删除子节点

OK，做个测试…… 

键入所有这些代码，加载页面，增加和删除一些即时贴。退出你的浏览器，再次加载，真正做一次演练！



现在可以删除即时贴了！



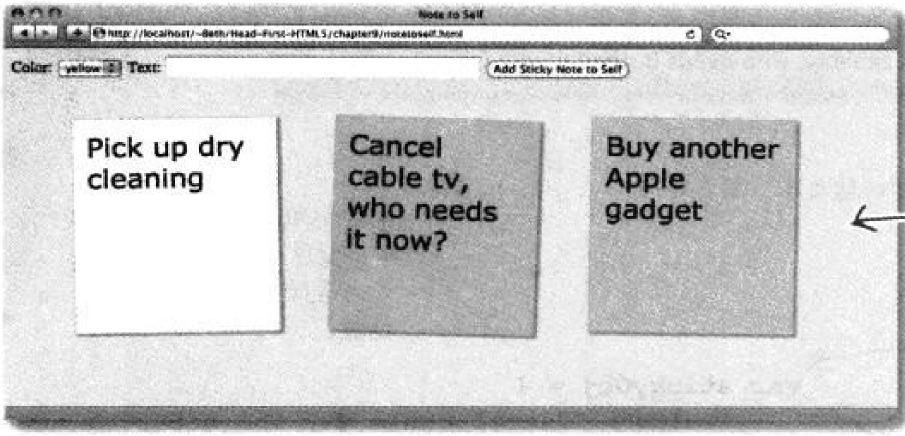
我们当然能做到！

来吧，根据你的经验，肯定能搞定。怎么做到呢？嗯，我们要创建一个对象来保存便条的文本和它的颜色，然后再存储这个对象，作为即时贴数据项的值，当然首先要使用JSON.stringify将它转换为一个字符串。

pin5i.com 拼吾爱

更新用户界面来指定颜色

目前，所有便条都是黄色的。如果能有一组即时贴颜色就好了。



可以在这里增加一个选择菜单，以便为便条选择你想要的颜色。

这样更好些，你觉得呢？

下面先来解决容易的部分：更新HTML，提供一个颜色选择菜单，可以从中选择颜色。编辑`notetoself.html`文件，更新表单，像下面这样增加颜色：

```
<html>
...
    <form>
        <label for="note_color">Color: </label>
        <select id="note_color">
            <option value="LightGoldenRodYellow">yellow</option>
            <option value="PaleGreen">green</option>
            <option value="LightPink">pink</option>
            <option value="LightBlue">blue</option>
        </select>
        <label for="note_text">Text:</label> <input type="text" id="note_text">
        <input type="button" id="add_button" value="Add Sticky Note to Self">
    </form>
...
</html>
```

我们只修改表单，其余的保持不变。

注意<select>的id，我们需要这个id在JavaScript中获取所选选项的值。

我们增加了4种即时贴颜色可供选择。

将为即时贴文本增加一个标签，让用户知道各个文本域分别用来做什么。

每个选项的值是一个颜色名，可以将它直接插入到即时贴的样式中。

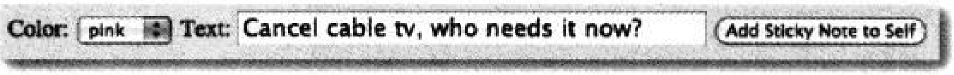
表单的其余部分保持不变。

我们一直在使用CSS定义便条的默认颜色，现在希望利用便条自身来保存它的颜色。所以，现在的问题是：如何把即时贴的颜色存储在localStorage中。

pin51.com 拼吾爱

JSON.stringify，不只是用于数组

除了即时贴文本，还要存储它的颜色，为此可以使用之前对stickiesArray采用的技术：可以将一个包含文本和颜色的对象作为即时贴的值存储在localStorage中。



取用户输入的颜色值和即时贴文本，把它们打包为一个简单的对象。



本地存储

在localStorage中存储这个对象，同时存储即时贴的键。

类似于stickiesArray，在调用localStorage.setItem保存值之前，必须对即时贴值调用JSON.stringify。

```
var stickyObj = {
    "value": "Cancel cable tv, who needs it now?",
    "color": "LightPink"
};
```

下面重写createSticky函数，除了存储即时贴文本还会同时存储颜色。为了表示文本和颜色，我们将使用前面提到的简便对象：

```
function createSticky() {
    var stickiesArray = getStickiesArray();
    var currentDate = new Date();
    var colorSelectObj = document.getElementById("note_color");
    var index = colorSelectObj.selectedIndex;
    var color = colorSelectObj[index].value;
    var key = "sticky_" + currentDate.getTime();
    var value = document.getElementById("note_text").value;
    var stickyObj = {
            "value": value,
            "color": color
    };
    localStorage.setItem(key, JSON.stringify(stickyObj));
    stickiesArray.push(key);
    localStorage.setItem("stickiesArray", JSON.stringify(stickiesArray));
    addStickyToDOM(key, stickyObj);
}
```

还是与往常一样，获取所选择的颜色选项的值。

然后使用这个颜色创建stickyObj：这个对象包含两个属性，即时贴的文本和用户选择的颜色。

将stickyObj放入localStorage之前，先对这个对象调用JSON.stringify。

现在，要向addStickyToDOM传入这个对象而不是一个文本串。这说明还需要更新addStickyToDOM，对不对？

pin51.com 拼吾爱 最新最全的下载站 分享资源站

使用新的stickyObj

既然要将stickyObj传入addStickyToDOM，所以我们还需要更新这个函数，要使用这个对象而不是之前传入的字符串，并设置即时贴的背景颜色。不过，这非常容易，下面就来看一看：

```
function addStickyToDOM(key, stickyObj) {
    var stickies = document.getElementById("stickies");
    var sticky = document.createElement("li");
    sticky.setAttribute("id", key);

    sticky.style.backgroundColor = stickyObj.color;

    var span = document.createElement("span");
    span.setAttribute("class", "sticky");
    span.innerHTML = stickyObj.value;
    sticky.appendChild(span);
    stickies.appendChild(sticky);
    sticky.onclick = deleteSticky;
}
```

这里需要把参数修改为stickyObj而不是即时贴的文本值。

从传入addStickyToDOM的stickyObj得到颜色。

注意，在JavaScript中设置背景颜色属性时，我们要指定backgroundColor，而不是像CSS中那样指定background-color。

HTML元素对象有一个style属性，可以用来访问这个元素的样式。

然后需要从这个对象得到将在即时贴中使用的文本值。

还有一个地方需要更新代码，这就是init函数，在这个函数中我们要从localStorage得到即时贴，并在首次加载页面时传递给addStickyToDOM。

```
function init() {
    var button = document.getElementById("add_button");
    button.onclick = createSticky;

    var stickiesAray = getStickiesArray();

    for (var i = 0; i < stickiesArray.length; i++) {
        var key = stickiesArray[i];
        var value = JSON.parse(localStorage[key]);
        addStickyToDOM(key, value);
    }
}
```

现在从localStorage获取即时贴的值时，需要调用JSON.parse对它解析，因为这是一个对象，而不再是一个字符串。

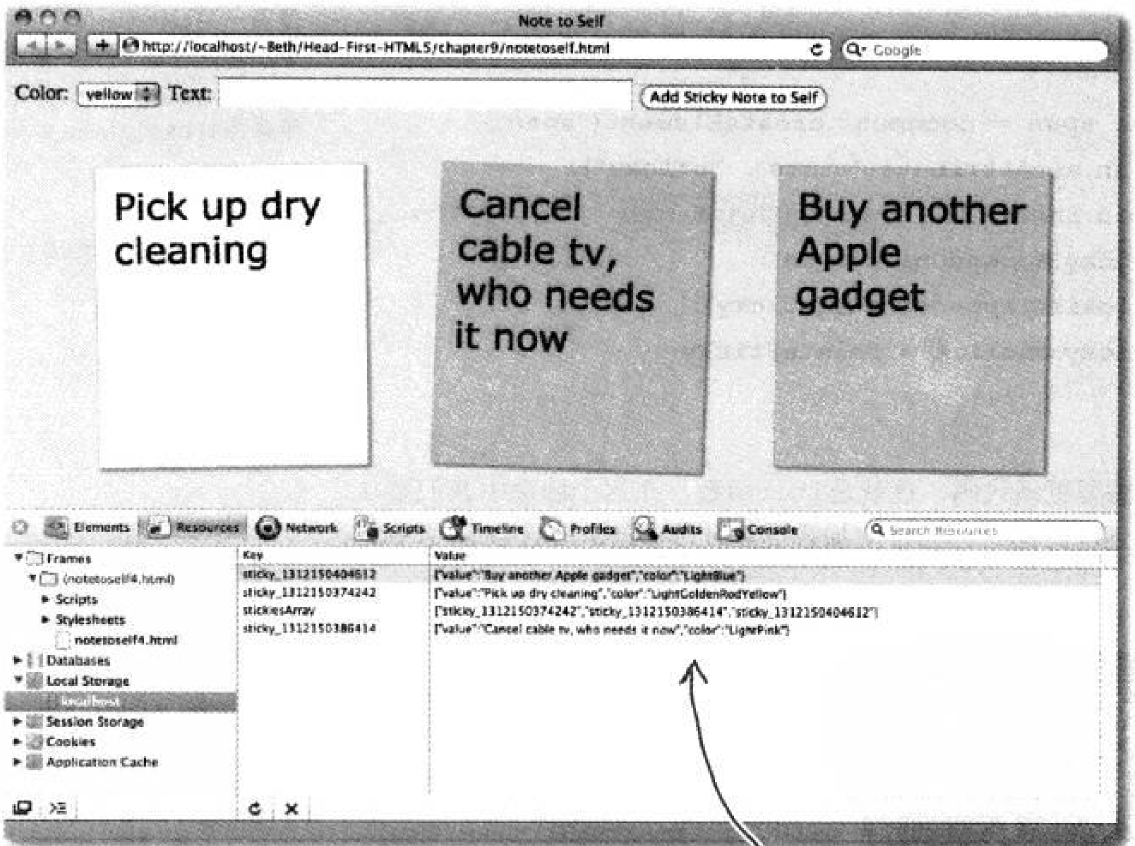
将这个对象传入addStickyToDOM而不再传入字符串（代码看上去是一样的，但是我们传入的东西不同）。

测试即时贴颜色

再次运行即时贴应用之前，首先需要清空你的localStorage，因为上一个版本的即时贴中没有存储颜色，现在我们会使用一种不同的格式来存储即时贴值。之前我们使用的是字符串，现在会使用对象。所以要清空你的localStorage，重新加载页面，再增加一些即时贴，为每个即时贴选择一个不同的颜色。以下是我们的即时贴（我们还会检查localStorage）：

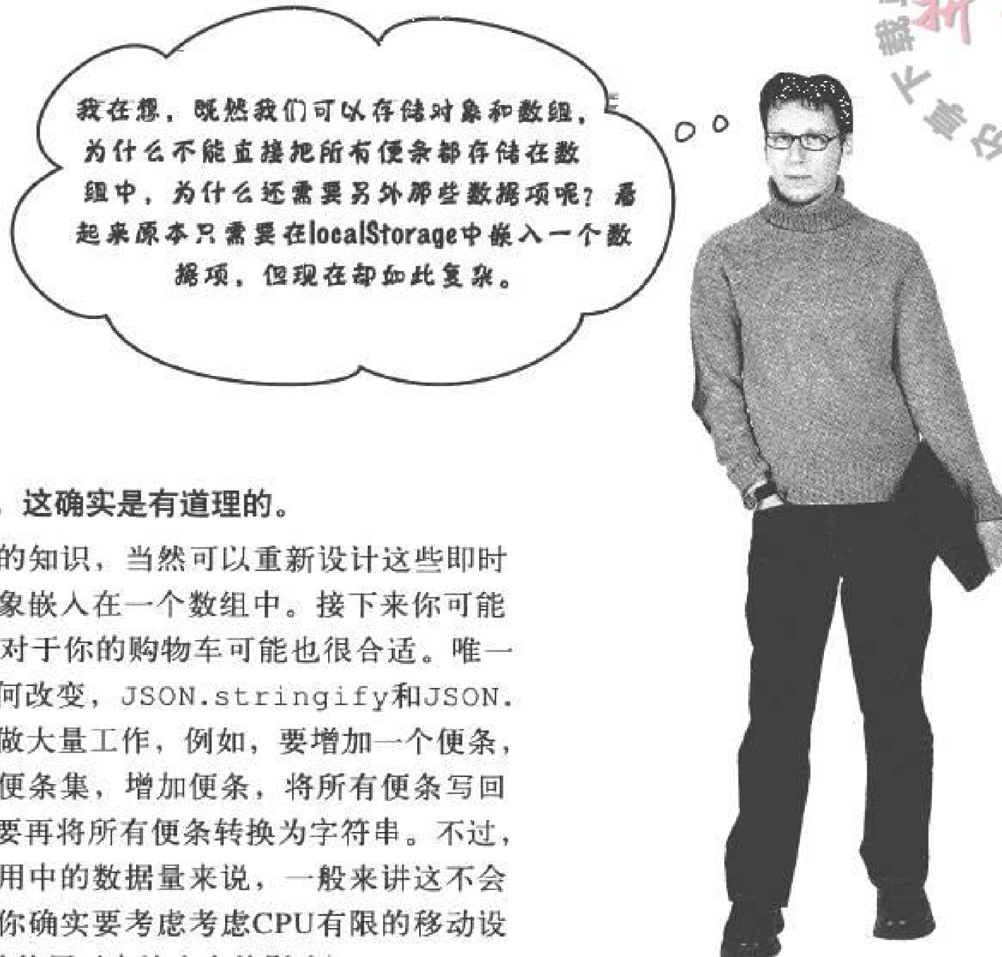
可以使用你的maintenance.html文件来清空localStorage，或者使用控制台来清空。

我们在增加即时贴时分别选择了黄色、粉红色和蓝色。



各个便条贴的值现在是一个对象（已经使用JSON转换为字符串），其中包含即时贴的文本值和这个即时贴的颜色。

pin51.com 拼吾爱



对于有些应用来说，这确实是有道理的。

根据我们现在掌握的知识，当然可以重新设计这些即时贴，让它们作为对象嵌入在一个数组中。接下来你可能就决定这样做。这对于你的购物车可能也很合适。唯一的缺点是只要有任何改变，`JSON.stringify`和`JSON.parse`方法就必须做大量工作，例如，要增加一个便条，我们必须解析整个便条集，增加便条，将所有便条写回到本地存储之前还要再将所有便条转换为字符串。不过，对于这个即时贴应用中的数据量来说，一般来讲这不会成为问题（不过，你确实要考虑考虑CPU有限的移动设备，还要考虑CPU的使用对电池寿命的影响）。

所以，是否把所有数据都打包到localStorage的一个对象或数组中，实际上这取决于你需要存储多少数据项，每个数据项有多大，以及你要对这些数据项做哪种类型的处理。

对于数量有限的即时贴来说，我们的实现可能有点过于“兴师动众”，不过它确实为我们提供了一个很好的途径来了解localStorage API，以及学习如何处理其中的数据项，这一点希望你同意。

pin51.com 拼吾爱

~~请不要~~自己尝试
(或者突破你的5MB)

我们已经告诉过你，每个用户的浏览器上会有总共5MB的存储空间，不过尽管5MB听上去很多，但要记住，所有数据都采用字符串格式存储，而不是采用一种字节高效的数据格式存储。对于一个很长的数（比如说国债），用浮点数形式表示时只占很少的存储空间，但如果采用字符串形式表示，所占的内存空间则会有几倍之多。所以，认识到这一点，5MB可能并不能像你原来以为的那样存储很多数据。

那么当你用完所有5MB时会发生什么？嗯，很遗憾，有些行为在HTML5规范中没有得到明确的定义，而超出5MB时浏览器的表现就属于这种没有明确定义的行为。如果你超出这个限制，不同的浏览器可能会有不同的举动—浏览器可能会询问你是否希望允许更多存储空间，也可能抛出一个QUOTA _ EXCEEDED _ ERR异常，可以这样捕获这个异常：

```
try {
    localStorage.setItem(myKey, myValue);
} catch(e) {
    if (e == QUOTA_EXCEEDED_ERR) {
        alert("Out of storage!");
    }
}
```

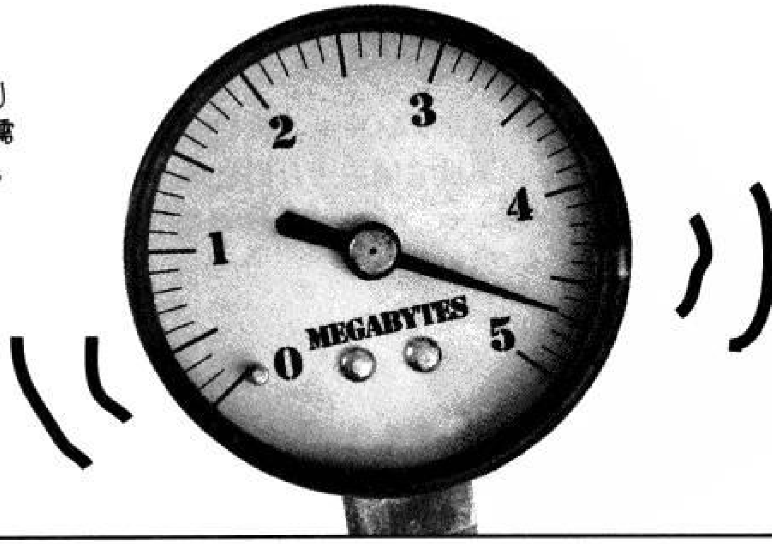
这里是try块中的一个setItem调用；如果出了问题，setItem抛出一个异常，就会调用catch块。

try/catch捕获try块中抛出的所有异常。

我们要查看这是不是一个存储配额错误（而不是另外某种异常）。如果是，就提醒用户。你可能想做些更有意义的处理，而不只是给出一个提醒。

这是我们还没有谈到的一个JavaScript领域，你可能想进一步了解这个内容。

目前并不是所有浏览器都抛出QUOTA_EXCEEDED_ERR异常。不过当你超出限制时，它们确实会抛出一个异常，所以可能需要处理设置数据项时出现异常的一般情况。





可以试着让浏览器发挥到极致，我们看不出有什么拒绝的理由，可以利用这个机件看看它的工作程度，看看它能达到的上限，另外看看在有压力的情况下它会有什么表现。下面来编写一些代码，让你的浏览器达到它的存储上限：

```
<html>
<head>
<script>

localStorage.setItem("fuse", "-");
while(true) {
    var fuse = localStorage.getItem("fuse");
    try {
        localStorage.setItem("fuse", fuse + fuse);
    } catch(e) {
        alert("Your browser blew up at" + fuse.length + " with exception: " + e);
        break;
    }
}
localStorage.removeItem("fuse");
</script>
</head>
<body>
</body>
</html>
```

开始时存储的字符串中只包括一个字符，键为"fuse"。

让它的长度一直增加……

……将这个串加倍（将它与自身连接）。

然后尝试将它写回到localStorage。

如果超出上限，我们的目的就达到了！此时会提醒用户，退出这个循环。

不要留下烂摊子，从localStorage删除数据项。

输入这个代码，然后加载，试试看吧！可以在不同的浏览器上尝试。

如果你有胆量运行这个代码，请在这里写出运行结果。

....

....

....



使用风险自负！

说实在的，这个代码可能会破坏你的浏览器，这可能导致你的操作系统出错，而这有可能导致你丢失之前所做的工作。所以使用时风险自负！

pin51.com 拼吾爱 最新最全资源分享下载站



不是卢克，但有另一位天行者[译注]。

看起来localStorage有一个姊妹，名叫sessionStorage。如果把使用`localStorage`的所有地方都替换为全局变量`sessionStorage`，数据项就只会在浏览器会话期间存储。所以，一旦会话结束（换句话说，用户关闭浏览器窗口时），本地存储中的数据项就会被删除。

`sessionStorage`对象支持的API与localStorage完全相同，所以没有更多新内容，要知道的你都已经知道了。

可以试试看！

译注：卢克·天行者（Luke Skywalker）是《星球大战》中银河系闻名的最伟大的英雄。

pin51.com 拼吾爱

WHO DOES WHAT?

你已经学完了localStorage API。下面你会看到这个API的所有主要角色以及相应的任务。看看你能不能确定谁做什么。我们已经帮你完成了一个。

clear

sessionStorage

key

setItem

removeItem

length

getItem

localStorage

QUOTA_EXCEEDED_ERR

用我可以长期存储数据项。

我拿到键和值，把它们写入localStorage。现在要记住，如果localStorage中已经有对应这个键的一个数据项，我不会警告你，而只是会覆盖它，所以你最好知道你要的是什么。

如果在localStorage中耽搁太久，使用了太多空间，你会得到一个异常，会受到我的批评。

需要删除一个数据项？我会很谨慎地完成这个任务。

只需要给我一个键，我会找出对应这个键的数据项，并把它的值交给你。

我属于短期的，只要你的浏览器打开，我就会存储你的内容。如果关闭浏览器，噗，你的所有内容都会消失无影踪。

如果你不再需要localStorage中的所有数据项，我会清除所有这些数据项，把它们扔掉，给你留下一个干干净净的空localStorage（要记住，我只能清空我自己的源）。

需要知道你的localStorage中有多少个数据项吗？找我吧。

交给我一个索引，我会告诉你localStorage中这个索引相应的键。

pin51.com 拼吾爱 最新最全资源分享下载站

既然你已经了解了localStorage，怎么使用呢？

localStorage有很多用法，即时贴应用就用到了localStorage，我们并不需要一个服务器，不过即使有服务器，localStorage也很有帮助。下面介绍另外几种使用localStorage的方法。







pin5i.com 拼吾爱

我有一个非常酷的游戏，它可以在两个不同的浏览器窗口同时工作，我在使用localStorage完成状态同步。





我要存储大量本地数据，让客户的应用能够在他们的移动设备上快速运行。在客户端能有一个超大的存储库，这对我来说实在是太漂亮了。

这为我提供了一种存储用户状态的新方法。我以前需要某种基于服务器端的会话和后备存储。现在可以直接在本地存储用户的状态，只是在必要的情况下才引入服务器端代码。



www.pin51.com 拼吾爱 最新最全电子书下载

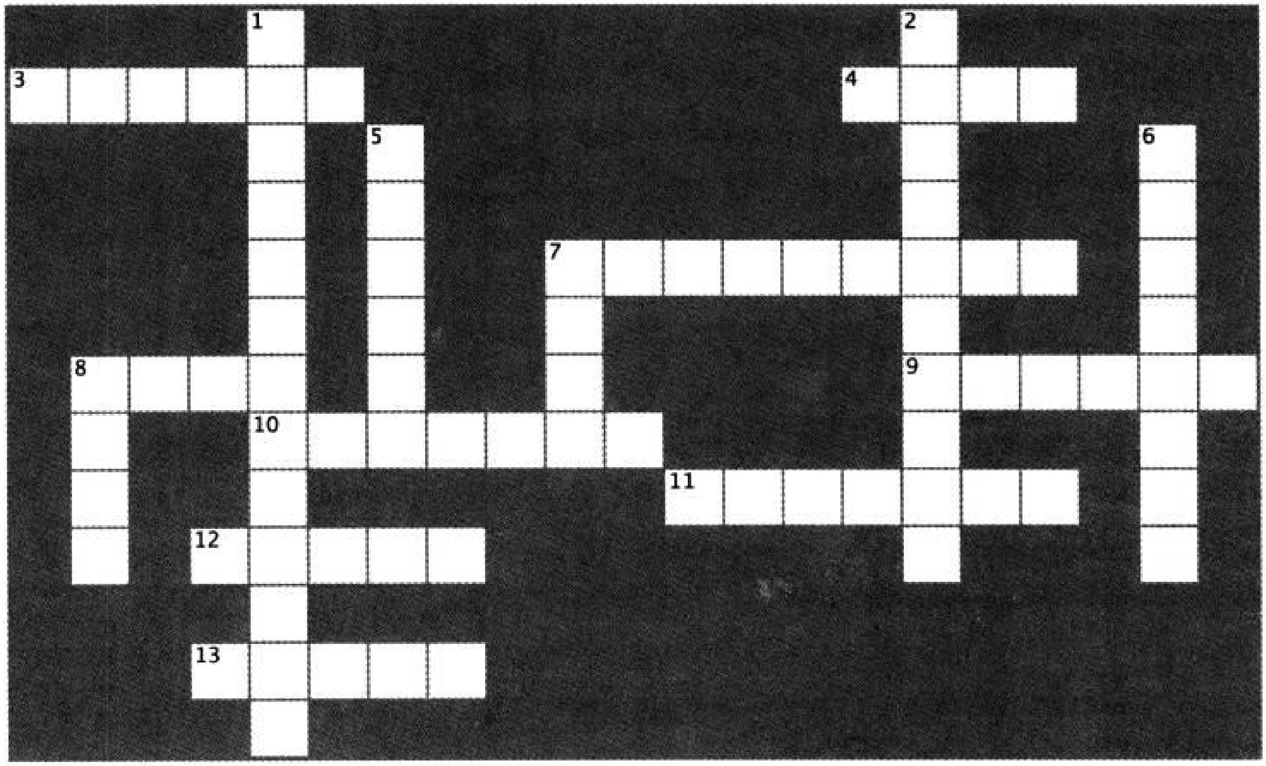
BULLET POINTS

- Web存储是浏览器中的一个存储库，也是一个API，可以用来从这个本地存储库保存和获取数据项。
- 大多数浏览器都为每个源提供了至少5MB的存储空间。
- Web存储包括本地存储和会话存储。
- 本地存储是持久的，即使你关闭浏览器窗口或退出浏览器，本地存储仍然保留。
- 会话存储中的数据项会在你关闭浏览器窗口或退出浏览器时删除。会话存储很适合临时数据项，而不是长期存储。
- 本地存储和会话存储使用完全相同的API。
- Web存储按源（可以认为是域）来组织。源就是Web上文档的位置（例如，wickedlysmart.com或headfirstlabs.com）。
- 每个域有一个单独的存储空间，所以存储在一个源的数据项对另一个源中的Web页面是不可见的。
- 使用localStorage.setItem(key)可以向本地存储增加一个值。
- 使用localStorage.getItem(key)可以从本地存储获取一个值。
- 可以使用与关联数组相同的语法在本地存储中设置和获取数据项。可以使用localStorage[key]来完成。
- 可以使用localStorage.key()方法列出localStorage中的键。
- localStorage.length是一个给定源localStorage中的数据项数。
- 使用浏览器中的控制台可以查看和删除localStorage中的数据项。
- 可以从localStorage直接删除数据项，只需右键点击一个数据项，并选择delete（注意，并非所有浏览器中都奏效）。
- 可以在代码中使用removeItem(key)方法和clear方法删除localStorage中的数据项。注意，clear方法会删除你完成这个清空操作时源所在localStorage中的所有内容。
- 每个localStorage数据项的键必须是唯一的。如果使用与一个现有数据项相同的键，会覆盖这个数据项的值。
- 要生成一个唯一的键，一种方法是使用当前时间（自1970年以来的毫秒数），并使用Date对象的getTime()方法。
- 要为你的Web应用创建一个合适的命名机制，使得即使数据项从本地存储删除，或者如果另一个应用在本地存储中创建了数据项，也能正常工作。
- Web存储目前只支持存储字符串作为对应键的值。
- 可以使用parseInt或parseFloat将localStorage中作为字符串存储的数字转换回真正的数字。
- 如果需要存储更复杂的数据，可以使用JavaScript对象，在存储之前会使用JSON.stringify把它们转换为字符串，在获取之后会使用JSON.parse再将其转换回对象。
- 本地存储对于移动设备尤其有用，可以用来降低带宽需求。
- 会话存储类似于本地存储，只不过浏览器存储库中保存的内容不会持久存储，如果你关闭标签页、窗口或退出浏览器，它们就不复存在。会话存储对于短期存储很有用，如购物会话。

pin51.com 拼吾爱 最新最全资源分享下载站

HTML5填字游戏

花点时间来测试你自己的本地存储。



横向

3. 我们使用localStorage的__________ 来创建键名时，会遇到一个问题：即时贴名之间存在间隔。
4. 路克天行者的姊妹。
7. 将对象存储在localStorage之前必须先__________对象。
8. 大多数浏览器为每个源提供_________ MB存储空间。
9. 通过查看事件__________可以检测用户点击了哪个即时贴。
10. 用这个方法在localStorage中存储一个数据项。
11. localStorage只能存储___________________。
12. 我们认为在localStorage中存储一个________可能是异想天开，不过事实上利用JSON，这真的可以做到。
13. 使用一个try/________来检测localStorage中的配额超出错误。

纵向

1. 我们使用 ______________来保存所有即时贴的键，从而很容易在localStorage中找到它们。
2. sessionStorage就像是localStorage，只不过不是___________，如果关闭浏览器窗口，它存储的数据就会消失。
5. 我们创建了 ________ 将即时贴文本和它的颜色存储在一个localStorage数据项中。
6. 使用 __________将一个字符串转换为整数。
7. Cookie存在一个__________ 问题。
8. 如果在你的浏览器中存储了一些内容，然后飞去_________，等你回来它依然在那里。



果壳游戏答案

想试试运气吗？或者是不是该说想试试你的水平吗？这里有一个游戏，可以测试一下你对localStorage的掌握程度，不过一定要专心。利用你目前掌握的在localStorage中获取和设置键/值对的知识，跟踪豌豆到底在哪个果壳下面。下面是我们的答案。

```
function shellGame() {
    localStorage.setItem("shell1", "pea");
    localStorage.setItem("shell2", "empty");
    localStorage.setItem("shell3", "empty");
    localStorage["shell1"] = "empty";
    localStorage["shell2"] = "pea";
    localStorage["shell3"] = "empty";
    var value = localStorage.getItem("shell2");
    localStorage.setItem("shell1", value);
    value = localStorage.getItem("shell3");
    localStorage["shell2"] = value;
    var key = "shell2";
    localStorage[key] = "pea";
    key = "shell1";
    localStorage[key] = "empty";
    key = "shell3";
    localStorage[key] = "empty";

    for (var i = 0; i < localStorage.length; i++) {
        var key = localStorage.key(i);
        var value = localStorage.getItem(key);
        alert(key + ": " + value);
    }
}
```

哪个果壳下面有豌豆？

键	值
shell1	empty
shell2	pea
shell3	empty

豌豆在shell2下面。



现在你的任务是更新所有代码，只要调用addStickyToDOM，都要传入键和值。

你应该已经更新了`init`和`createSticky`中的所有addStickyToDom调用，如下：

```
addStickyToDOM(key, value);
```

pin51.com 拼吾爱

Sharpen your pencil Solution

你认为我们当前的实现在哪些方面可能会有问题，请在旁边打勾：

- [x] 如果localStorage中有很多数据项不是即时贴，显示即时贴会很低效。
- [x] 如果另一个应用删除它自己的数据项而导致localStorage长度变小时，即时贴可能会被setItem覆盖。
- [x] 很难很快说出到底有多少个即时贴，必须迭代处理localStorage中的每一个数据项来得到所有即时贴。
- [ ] 用Cookie吧，肯定比解决这些问题更容易！

Exercise Solution

还需要确定具体如何在localStorage中存储一个数组。

你可能已经猜到，我们可以使用JSON来创建数组的一个字符串表示，如果是这样，那你猜对了。一旦想到这一点，就可以在localStorage中存储数组了。

应该记得，JSON API中只有两个方法：stringify和parse。下面具体运用这两个方法来完成init函数：

```
function init() {
    // button code here...
    var stickiesArray = localStorage["stickiesArray"];
    if (!stickiesArray) {
        stickiesArray = [];
        localStorage.setItem("stickiesArray", JSON.stringify(stickiesArray));
    } else {
        stickiesArray = JSON.parse(stickiesArray);
    }
    for (var i = 0; i < stickiesArray.length; i++) {
        var key = stickiesArray[i];
        var value = localStorage[key];
        addStickyToDOM(value);
    }
}
```

从localStorage获取数组。

如果localStorage中没有，就创建一个空数组，并把它赋给变量stickiesArray。此时变量stickiesArray是一个字符串。

如果必须创建一个新数组，要使用JSON.stringify创建数组的一个字符串表示，然后存储……

如果即时贴数组已经存储在localStorage中（作为一个字符串），则需要使用JSON进行解析。解析之后将键数组赋给stickiesArray变量。

再明确一下，我们首先得到stickiesArray指向的字符串，将它解析为一个数组，然后再把这个数组赋给stickiesArray变量。

pin51.com 拼吾爱 最新最全最优秀的资源下载站

~~请不要~~自己尝试
(或者突破你的5MB)

我们已经告诉过你，每个用户的浏览器上会有总共5MB的存储空间，不过尽管5MB听上去很多，但要记住，所有数据都采用字符串格式存储，而不是采用一种字节高效的数据格式存储。对于一个很长的数（比如说国债），用浮点数形式表示时只占很少的存储空间，但如果采用字符串形式表示，所占的内存空间则会有几倍之多。所以，认识到这一点，5MB可能并不能像你原来以为的那样存储很多数据。

那么当你用完所有5MB时会发生什么？嗯，很遗憾，有些行为在HTML5规范中没有得到明确的定义，而超出5MB时浏览器的表现就属于这种没有明确定义的行为。如果你超出这个限制，不同的浏览器可能会有不同的举动——浏览器可能会询问你是否希望允许更多存储空间，也可能抛出一个QUOTA _ EXCEEDED _ ERR异常，可以这样捕获这个异常：

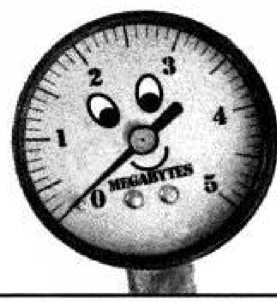
```
try {
    localStorage.setItem(myKey, myValue);
} catch(e) {
    if (e == QUOTA_EXCEEDED_ERR) {
        alert("Out of storage!");
    }
}
```

这里是try块中的一个setItem调用。如果出了问题，setItem抛出一个异常，就会调用catch块。

try/catch捕获try块中抛出的所有异常。

我们要查看这是不是一个存储配额错误（而不是另外某种异常）。如果是，就提醒用户。你可能想做些更有意义的处理，而不只是给出一个提醒。

目前并不是所有浏览器都抛出QUOTA_EXCEEDED_ERR异常。不过当你超出限制时，它们确实会抛出一个异常，所以可能需要处理设置数据项时出现异常的一般情况。





可以试着让浏览器发挥到极致，我们看不出有什么拒绝的理由，可以利用这个机件看看它的工作程度，看看它能达到的上限，另外看看在有压力的情况下它会有什么表现。下面来编写一些代码，让你的浏览器达到它的存储上限：

```
<html>
<head>
<script>

localStorage.setItem("fuse", "-");
while(true) {
    var fuse = localStorage.getItem("fuse");
    try {
        localStorage.setItem("fuse", fuse + fuse);
    } catch(e) {
        alert("Your browser blew up at" + fuse.length + " with exception: " + e);
        break;
    }
}
localStorage.removeItem("fuse");
</script>
</head>
<body>
</body>
</html>
```

开始时存储的字符串中只包括一个字符，键为"fuse"。

让它的长度一直增加……

……将这个串加倍（将它与自身连接）。

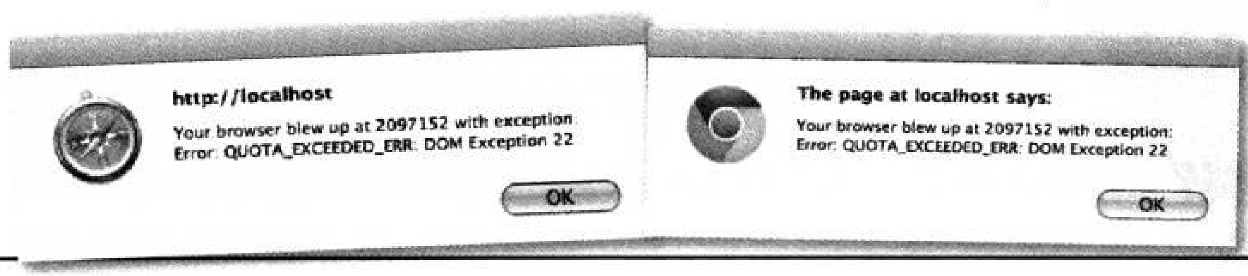
然后尝试将它写回到localStorage。

如果超出上限，我们的目的就达到了！此时会提醒用户，退出这个循环。

不要留下烂摊子，从localStorage删除数据项。

输入这个代码，然后加载，试试看吧！可以在不同的浏览器上尝试。

在Safari和Chrome上的运行结果。



pin51.com 拼吾爱

WHO DOES WHAT? 答案

你已经学完了localStorage API。下面你会看到这个API的所有主要角色以及相应的任务。看看你能不能确定谁做什么。我们已经帮你完成了一个。

clear

sessionStorage

key

setItem

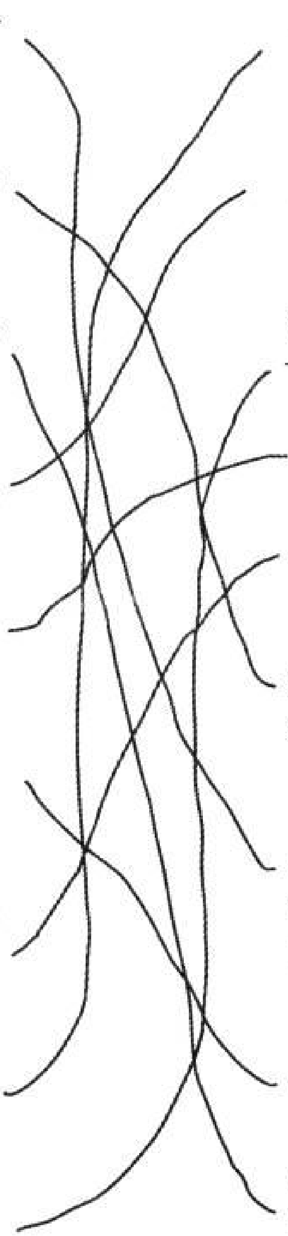
removeItem

length

getItem

localStorage

QUOTA_EXCEEDED_ERR



用我可以长期存储数据项。

我拿到键和值，把它们写入localStorage。现在要记住，如果localStorage中已经有对应这个键的一个数据项，我不会警告你，而只是会覆盖它，所以你最好知道你要的是什么。

如果在localStorage中耽搁太久，使用了太多空间，你会得到一个异常，会受到我的批评。

需要删除一个数据项？我会很谨慎地完成这个任务。

只需要给我一个键，我会找出对应这个键的数据项，并把它的值交给你。

我属于短期的，只要你的浏览器打开，我就会存储你的内容。如果关闭浏览器，噗，你的所有内容都会消失无影踪。

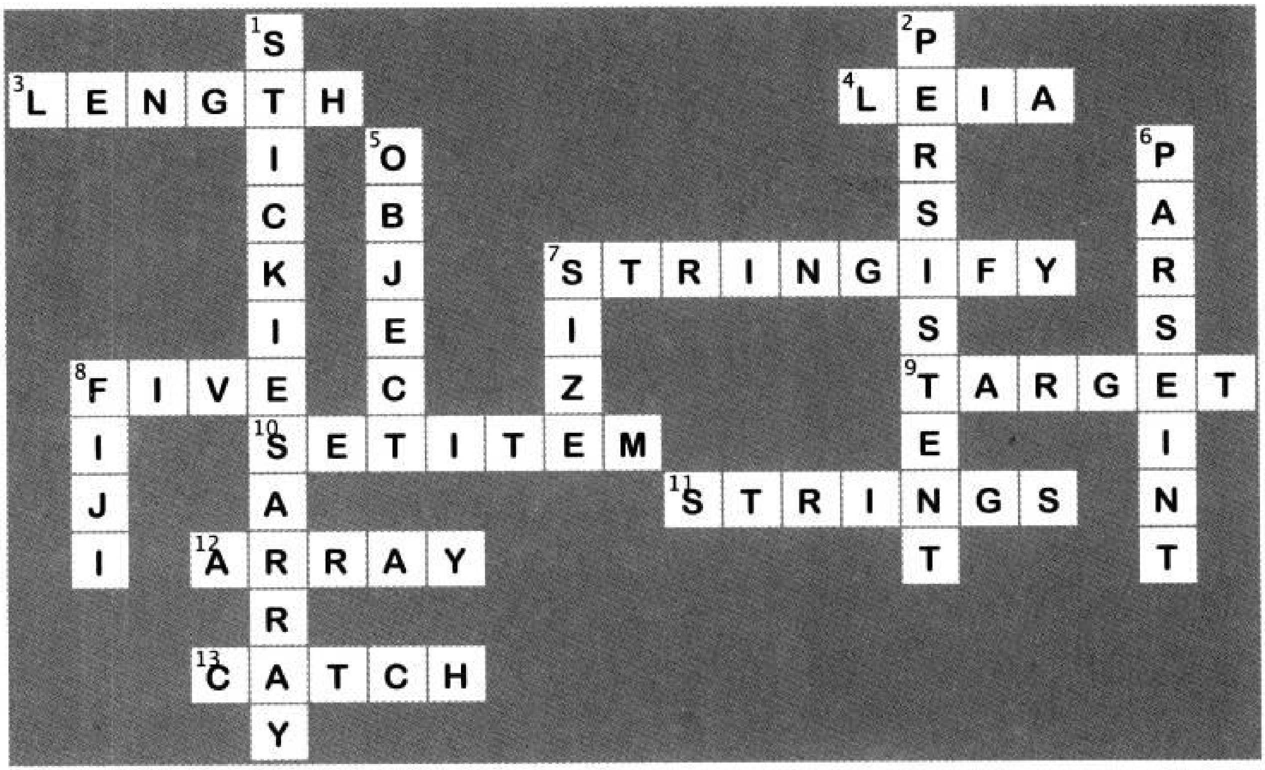
如果你不再需要localStorage中的所有数据项，我会清除所有这些数据项，把它们扔掉，给你留下一个干干净净的空localStorage（要记住，我只能清空我自己的源）。

需要知道你的localStorage中有多少个数据项吗？找我吧。

交给我一个索引，我会告诉你localStorage中这个索引相应的键。

pin51.com 拼吾爱

HTML5填字游戏答案



pin51.com 拼吾爱 最新编程资源分享下载站

10 运用JavaScript

Web工作线程



脚本运行缓慢，你还想继续运行吗？ 如果你使用JavaScript或浏览Web的时间足够长，可能见过“slow script”消息告诉你脚本运行缓慢。不过，既然你的新机器里已经拥有那些多核处理器，脚本怎么可能会运行得慢呢？这是因为，JavaScript一次只能做一件事。不过，有了HTML5和Web工作线程，一切都改变了。现在完全可以创建你自己的JavaScript工作线程来完成更多工作。也许你只是想设计一个更有响应性的应用，或者可能只想最大限度地发挥机器的CPU能力，不论怎样，Web工作线程都竭诚为你提供帮助。带上你的JavaScript经理帽，找些工人干活吧！

pin51.com 拼吾爱 最新最全电子书免费下载站

可怕的“slow script”

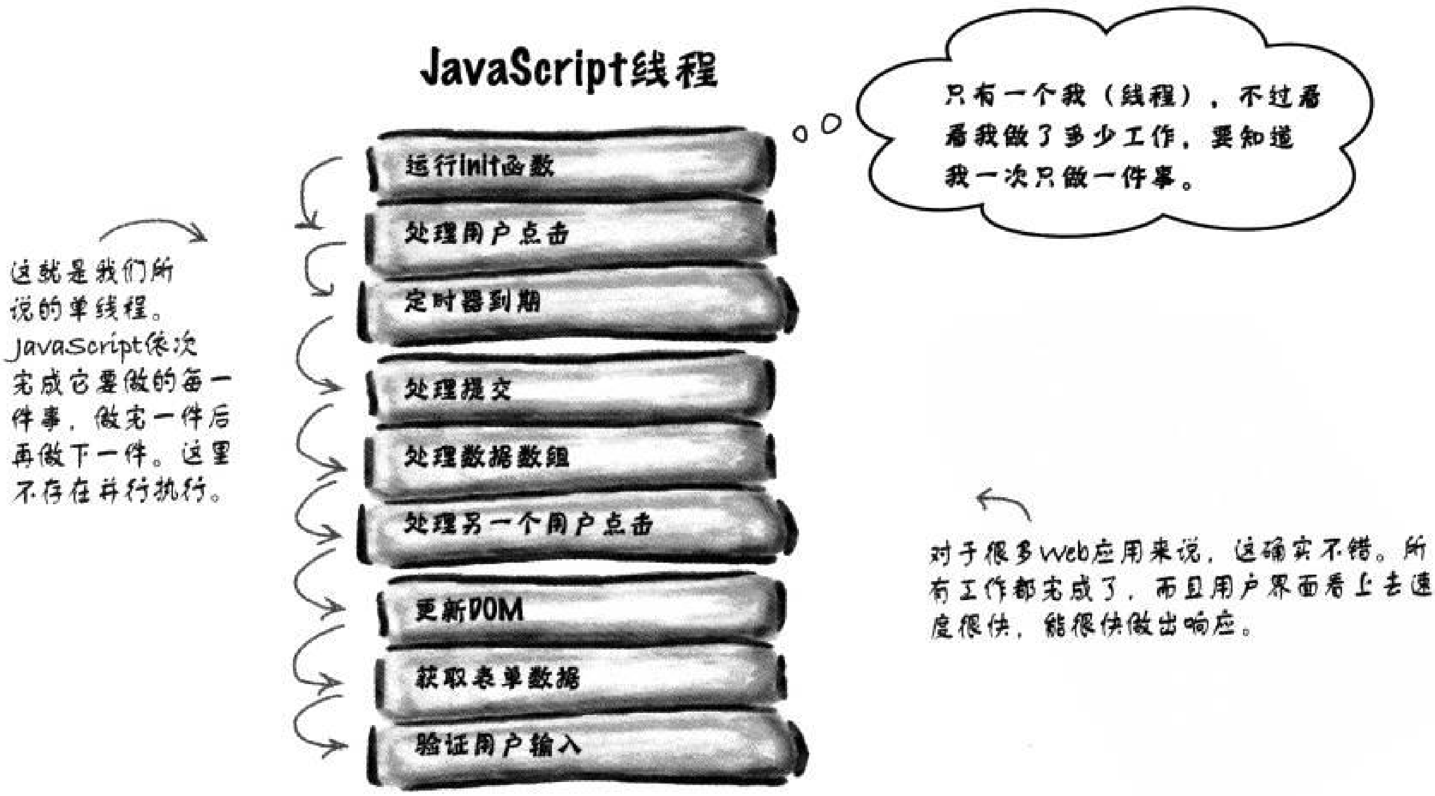
JavaScript有很多好处，其中有一条：它一次只做一件事情。这就像我们所说的“单线程”。为什么这算一件好事呢？因为这样一来，编程就很简单。倘若同时执行大量线程，要想编写一个能正确工作的程序可能很有难度。

单线程也有缺点，如果你交给一个JavaScript程序太多的工作，它就会受不了，最后我们可能会收到一个“slow script”对话框，告诉你脚本运行缓慢。只运行一个线程还有另外一个限制，如果JavaScript代码在很努力地工作，会占用大量计算能力，对你的用户界面或用户交互来说就所剩无几了，应用看上去会相当迟缓，甚至毫无响应。



JavaScript如何分配时间

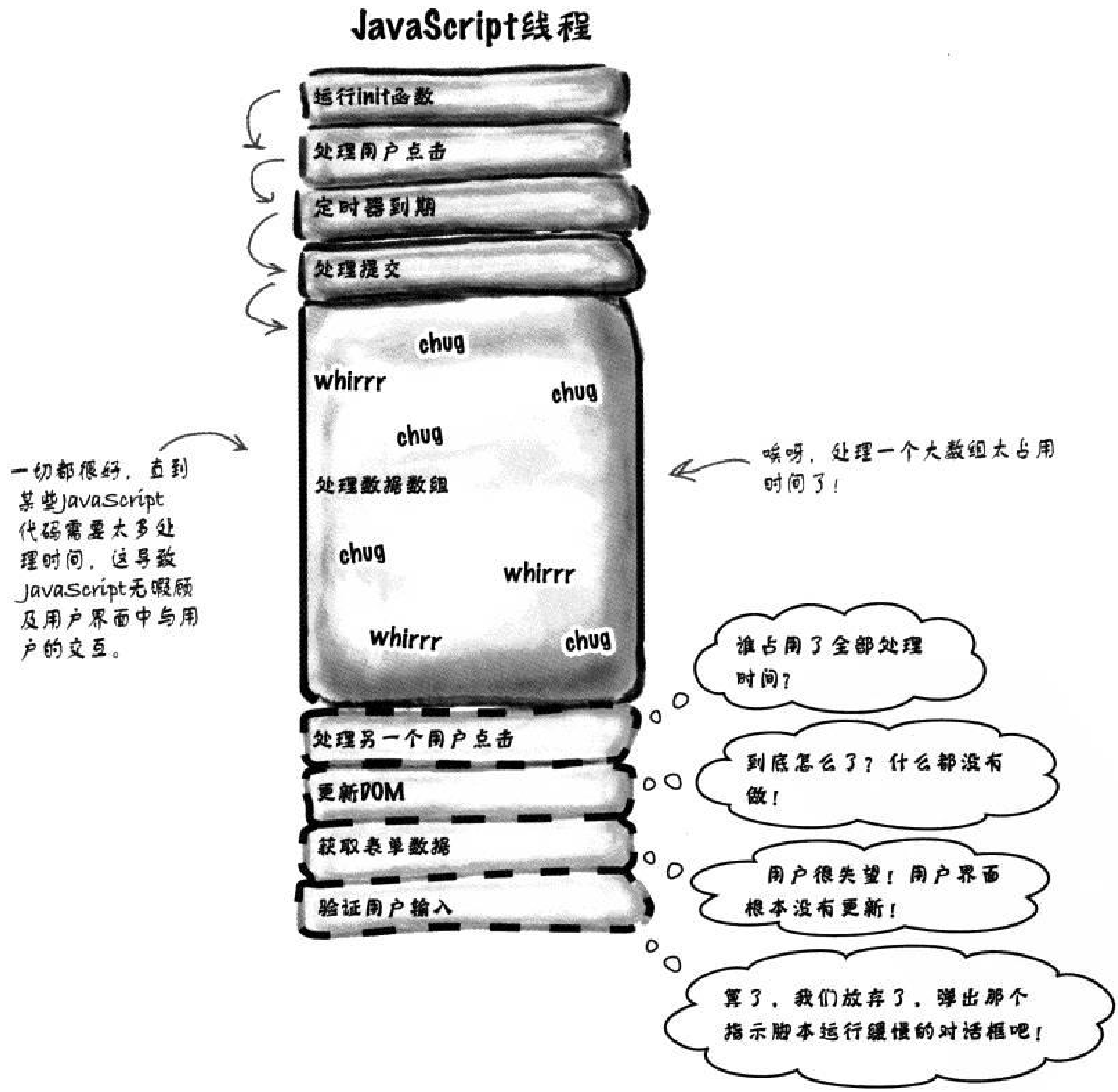
下面来看JavaScript如何处理一个典型页面的任务，通过这个分析来了解前面所说的是什么意思：



www.pin51.com 拼吾爱 最新最全的资源分享下载站

单线程遇到麻烦

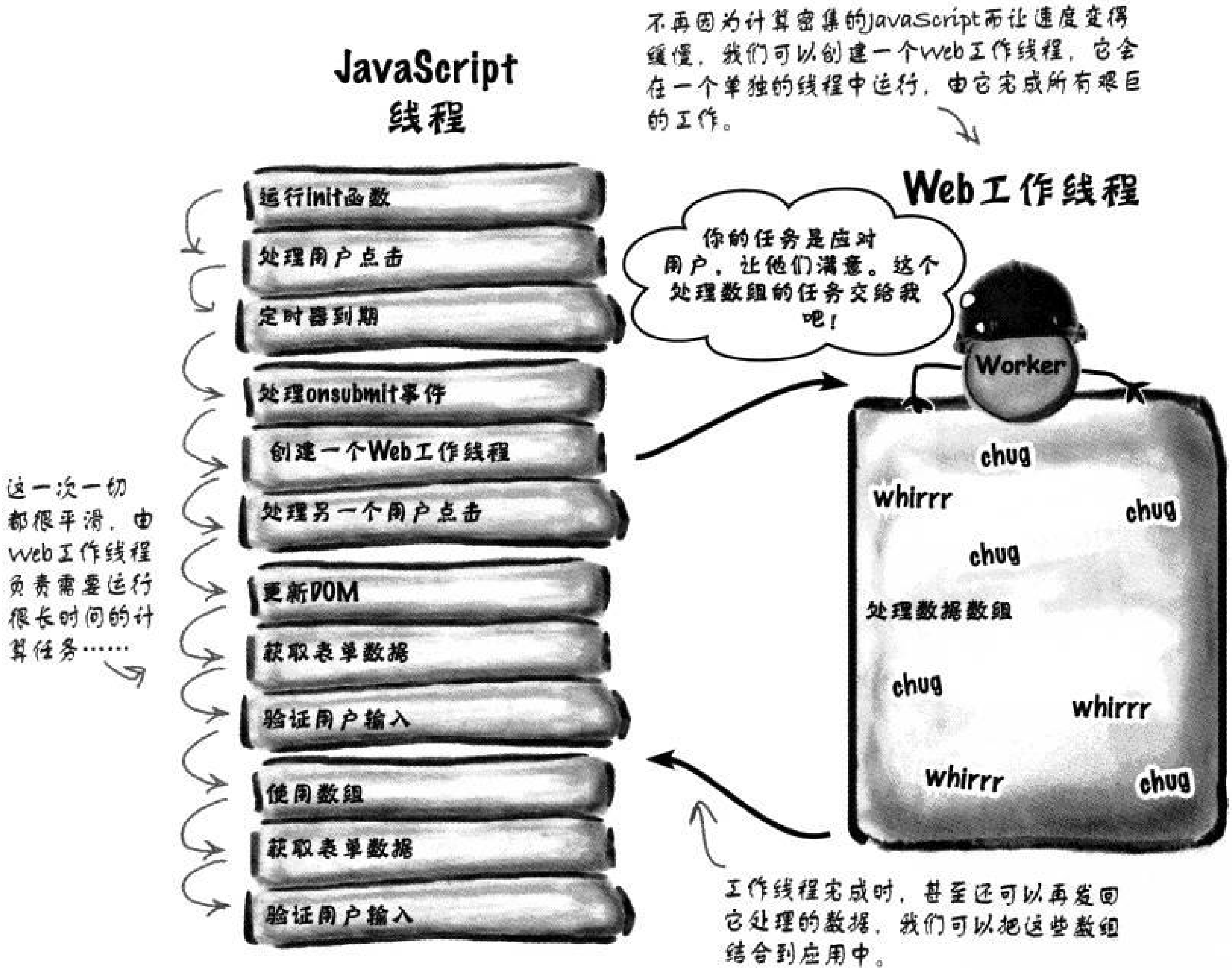
对于很多应用来说，JavaScript的这个单线程计算模式确实表现不错，这一点不假，而且正如我们所说，采用这种模式，编程会很简单。不过，编写“计算密集”的代码时，就会影响JavaScript的能力，以至于不能有效地完成所有工作，这个单线程模型就有问题了。



pin51.com 拼吾爱 最新最全的下载站 分享资源

增加另一个控制线程提供帮助

在HTML5之前，我们的页面和应用中只能有一个控制线程，但是有了Web工作线程，现在终于有办法创建另外的控制线程来提供帮助了。所以，如果有一些代码的执行需要花费太长时间，就可以创建一个Web工作线程，由它处理这个任务，而让主JavaScript控制线程确保与浏览器和用户的交互一切正常。



前面说过，一个控制线程可以让编程简单而容易，而现在我们的做法似乎偏离了这种说法，这一点不可否认。不过，后面你就会了解到，Web工作线程经过了精心设计，对于程序员来说，一切仍然很简单、很容易，而且一样安全。稍后就来分析这是如何做到的……



JavaScript（再次）闪亮登场

本周访谈：
JavaScript把时间都用在哪里了。

Head First: 欢迎回来，JavaScript，你能来太棒了。

JavaScript: 我很乐意接受采访，只要与我的日程安排不冲突，要知道我有太多的事情要做。

Head First: 这正是我考虑的今天讨论的主题。你实在太成功了，有那么多业务，你是怎么一一做到的呢？

JavaScript: 嗯，我有我的哲学，一次只做一件事，要做就把它做好。

Head First: 怎么会一次只做一件事情？在我们看来，你一边在获取数据，一边显示页面，同时还与用户交互，另外还没忘记管理定时器和提醒，等等，等等……

JavaScript: 没错，这些都是我做的，不过不论我做什么，我都只做那一件事。所以，如果我在处理与用户的交互，在我完成之前，我所做的就只是处理用户交互。

Head First: 这怎么可能呢？如果一个定时器到期了，或者有网络数据到达，或者发生了其他情况，难道你不停下来处理吗？

JavaScript: 出现一个事件时，比如说你刚才提到的，这个事件会增加到一个队列中。在我完成当前正在做的工作之前，我甚至看都不看它一眼。这样一来，我就能保证正确、安全，而且高效地完成所有工作。

Head First: 那么，你处理队列中的某个任务会不会太迟啊？

JavaScript: 哦，这是有可能的。不过，幸运的是，我是浏览器Web页面后台的技术，所以即使我有一点滞后又有什么关系呢？在这方面，你应该与那些运行航天器推进器或核电站控制器代码的家伙聊一聊，他们有不同的处世原则—这也是他们能挣大钱的原因。

Head First: 我一直想知道在浏览器上收到“Slow script, do you want to continue”（脚本运行缓慢，是否希望继续）对话框时到底发生了什么？是你休息了吗？

JavaScript: 休息？哈，真是开玩笑。那是因为有人建立的页面让我有太多的工作要做，我根本做不完！如果你写了一段JavaScript占用了我的全部时间，这就会影响你与用户的交互。我已经尽我所能了。

Head First: 听上去你需要些帮助。

JavaScript: 好在有了HTML5，现在我有了帮手，正是因此才引入了Web工作线程。如果你需要编写计算密集的代码，可以使用Web工作线程分担一些工作。这样一来，我还能继续关注我的工作，由工作线程帮助完成一些繁重的任务（而且不会干扰我）。

Head First: 有意思，下面我们就来看一看。下一个问题…… 噢，等一下，他居然走了，看来他得完成他的下一个任务了。真是个刻板的家伙，是吧？

pin51.com 拼吾爱 最新最全资源分享下载站

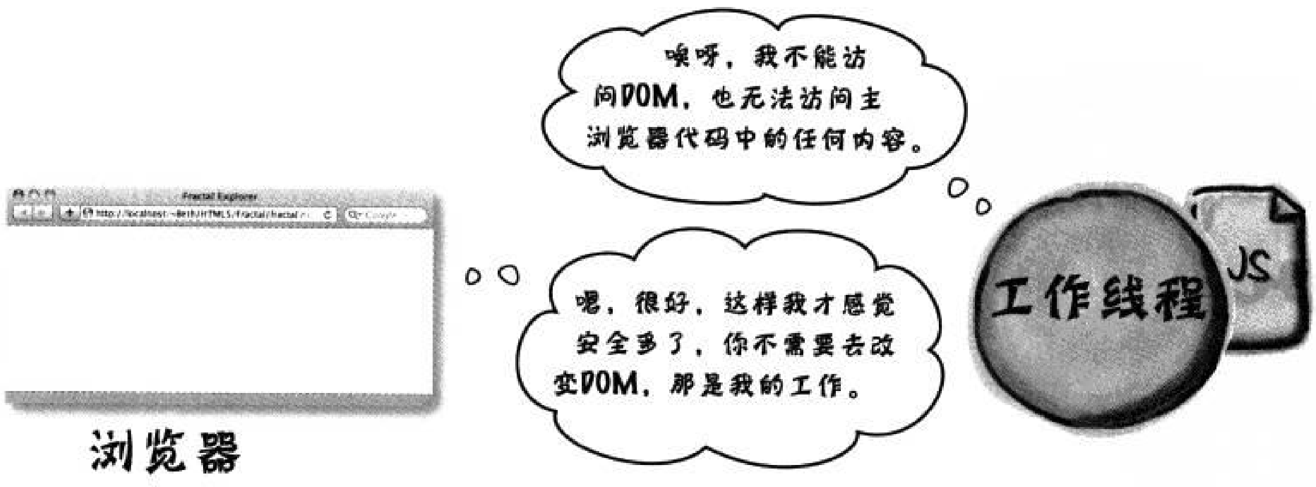
Web工作线程如何工作

下面来看Web工作线程生命中的一天如何渡过：工作线程如何创建，如何知道要做什么，另外如何将结果返回给主浏览器代码。

要使用Web工作线程，浏览器首先必须创建一个或多个工作线程来帮助完成计算任务。每个工作线程都由各自的JavaScript文件定义，其中包含完成工作所需的全部代码（或代码的引用）。

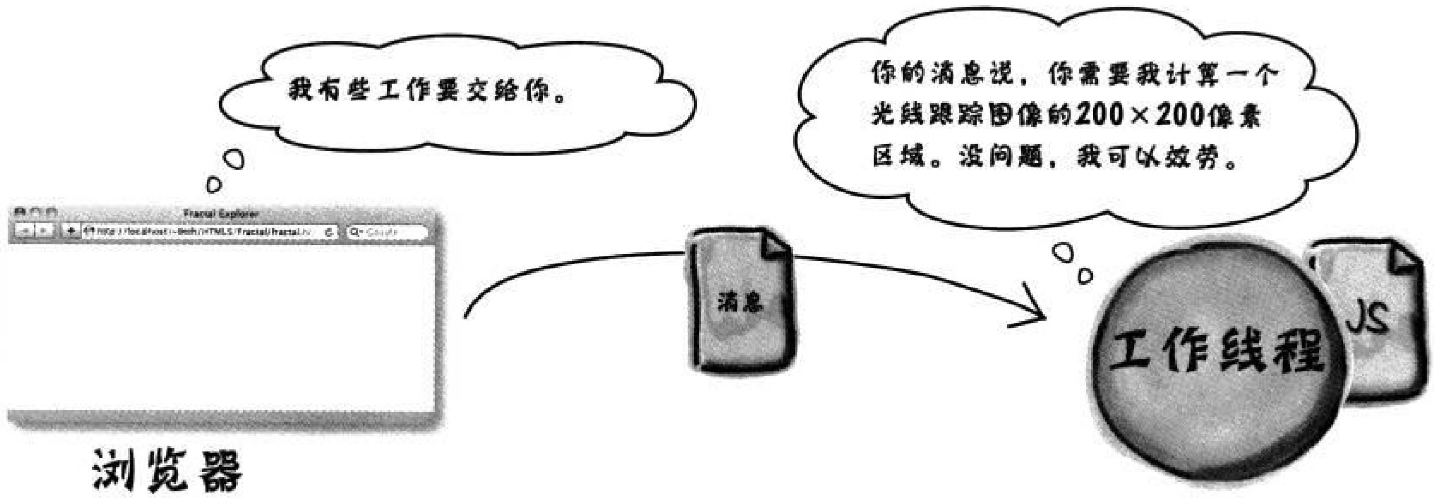


工作线程生活在一个相当受限的世界中。它们无法访问主浏览器代码能够访问的很多运行时对象，如DOM或主代码中的所有变量或函数。

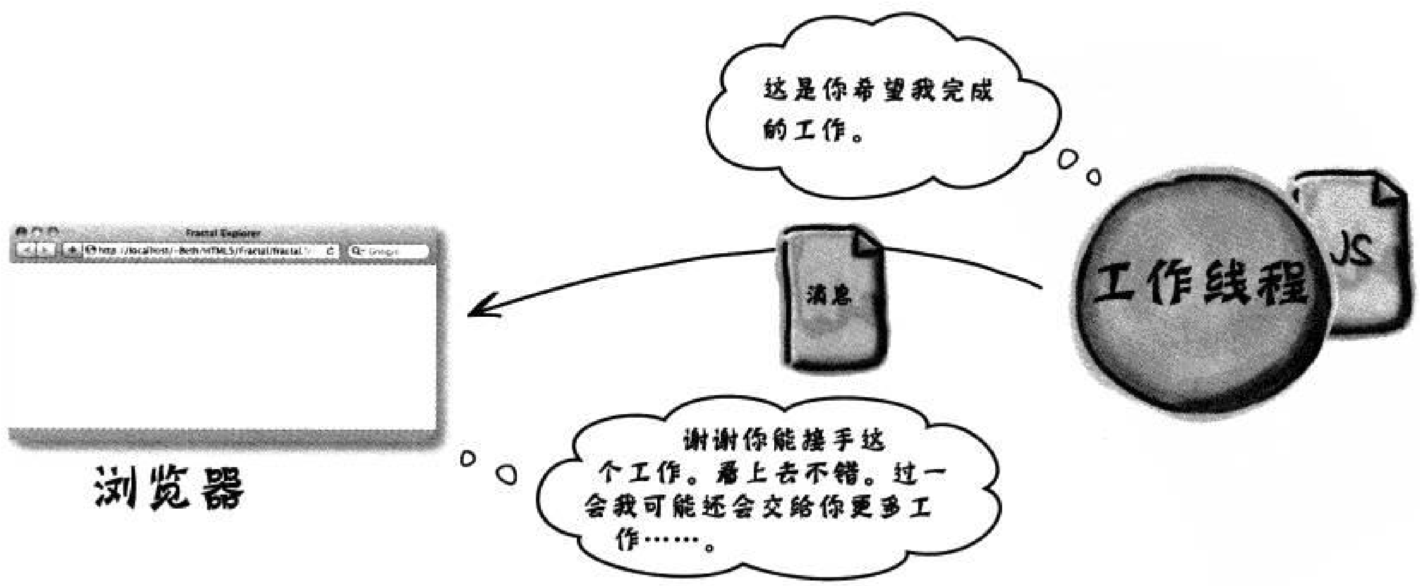


pin5i.com 拼吾爱 最新编程资源分享下载站

要让一个工作线程开始工作，浏览器通常会向它发送一个消息。工作线程代码接收到这个消息，查看其中是否有特殊的指令，然后开始工作。



工作线程完成它的工作时，会发回消息，并提供它处理的最终结果。
主浏览器代码会得到这些结果，把它们以某种方式结合到页面中。

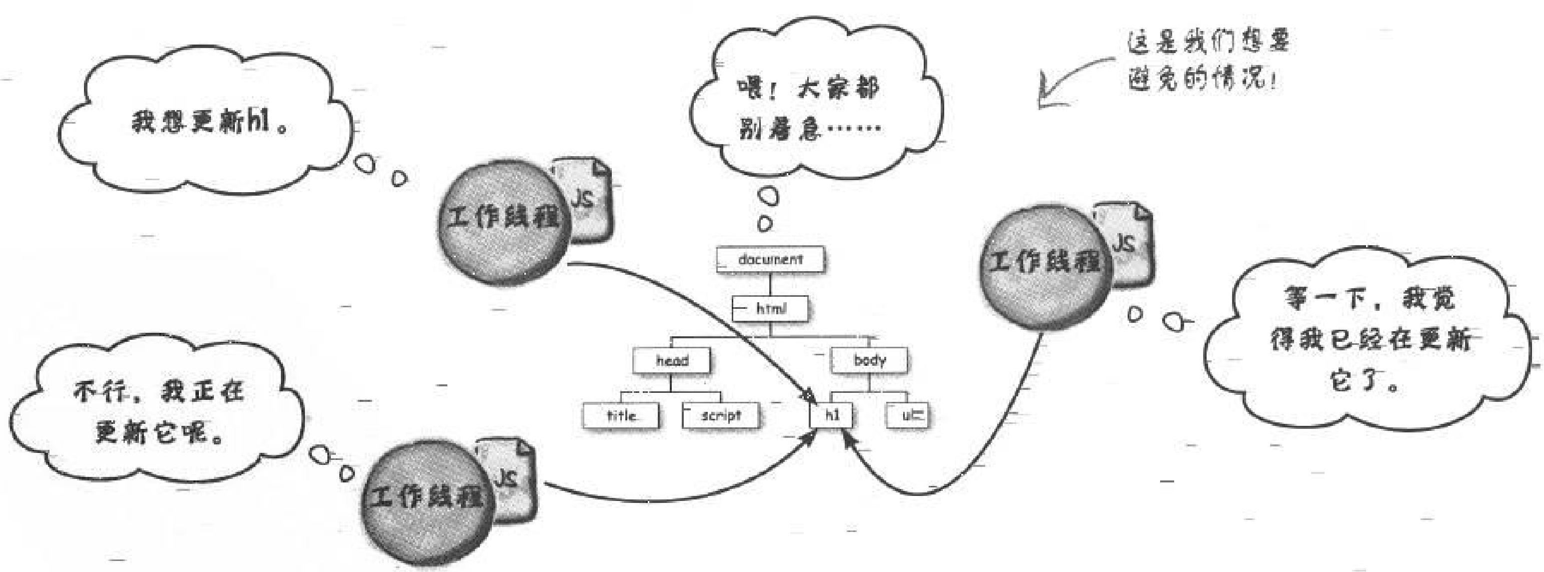


pin51.com 拼吾爱



这是为了保证高效。

DOM和JavaScript之所以如此成功，很大的一个原因就是我们能高度优化DOM操作，因为只有一个线程能访问DOM。如果让多个计算线程并发地改变DOM，就会严重影响它的性能（而且实现浏览器的人必须花很大功夫来确保对DOM的修改是安全的）。事实上，如果允许同时对DOM做大量修改，很容易导致DOM处于一种不一致状态的情况，这可不好，非常糟糕。



最新编程资源分享下载站 拼吾爱 pin5i.com

Sharpen your pencil

下面给出工作线程的所有可能的用法。你认为哪些可以改善应用的设计和性能？

- [ ] 缓存数据，以便在页面中使用。
- [ ] 处理数组中的大量数据或者来自Web服务的庞大JSON响应。
- [ ] 管理数据库连接，并为主页面增加和删除记录。
- [ ] 自动赌马代理。
- [ ] 分析视频。
- [ ] ……………………………………
- [ ] ……………………………………
- [ ] ……………………………………
- [ ] 用户键入时对页面完成拼写检查。
- [ ] 轮询Web服务，发生特殊情况时提醒主页面。
- [ ] 画布中数据的图像处理。
- [ ] 突出显示代码语法或其他语言。
- [ ] 根据用户正在做的工作预获取数据。
- [ ] 管理你的页面的广告宣传。
- [ ] ……………………………………
- [ ] ……………………………………
- [ ] ……………………………………

可以在这里写出你的想法！

所有答案都是可行的用法，不过你可能对其中两个有争议：拼写检查与语法检查在主页面代码中完成可能更合适；另外最好根本不要赌马。☺

pin51.com 最新 拼吾爱

Watch it!

Google的Chrome浏览器有一些额外的安全限制，不允许直接从文件运行Web工作线程。如果试图这么做，页面不会运行，你也得不到任何提示告诉你页面为什么不运行（而且没有任何错误消息通知你已经出错）。

所以，要运行这一章的例子，建议你要么使用另外一个浏览器，要么运行你自己的服务器，并从http://localhost运行。或者如果可以访问一个托管服务器，也可以把这些例子上传到托管服务器运行。

还可以使用Chrome运行时开关，allow-file-access-from-files。

不过，除非为了测试代码，否则我们不建议使用这个开关。

Watch it!

几乎所有现代浏览器都支持Web工作线程，不过有一个例外：Internet Explorer 9。好在还有一个好消息：IE10以及以后版本将支持Web工作线程，但是对于IE9和之前的所有IE版本，则无法提供这种体验。

不过，不用将IE单做考虑，用下面的方法可以很容易地查看一个浏览器是否支持Web工作线程：

如果支持工作线程，全局作用域window中会定义有属性Worker。

如果没有定义Worker，说明这个浏览器未提供支持。

```
if (window["Worker"]) {
    var status = document.getElementById("status");
    status.innerHTML = "Bummer, no Web workers";
}
```

你可能会以适用于具体应用的某种方式处理这种情况。在这里，我们只是在id="status"的元素中放入一个消息，让用户知道浏览器不支持Web工作线程支持。

www.pin51.com 拼吾爱 最新最全大型电子书分享站

第一个Web工作线程……

下面来具体创建一个工作线程，看看它是如何工作的。为此，我们需要一个页面放入所有内容。先从最简单的HTML5标记开始，把下面的代码键入**pingpong.html**：

带上安全帽，出发吧。只需要给我一个JavaScript文件，告诉我你希望我做什么。

```
<!doctype html>
<html lang="en">
  <head>
    <title>Ping Pong</title>
    <meta charset="utf-8">
    <script src="manager.js"></script>
  </head>
<body>
  <p id="output"></p>
</body>
</html>
```

这个JavaScript代码将创建和管理所有工作线程。

我们会在这里放入工作线程的一些输出。



Web工作线程。

如何创建Web工作线程

开始实现`manager.js`之前，下面先来看如何具体创建一个Web工作线程：

为了创建新的工作线程，我们创建了一个新的Worker对象……

```
var worker = new Worker("worker.js");
```

将这个新工作线程赋给一个名为worker的JavaScript变量。

……"worker.js" JavaScript文件中包含这个工作线程的相应代码。

这样就能创建一个工作线程了，当然，你不必就此止步，只要愿意，还可以创建更多的工作线程：

```
var worker2 = new Worker("worker.js");
var worker3 = new Worker("worker.js");

var another_worker = new Worker("another_worker.js");
```

可以很容易地创建另外两个工作线程，它们使用的代码与第一个工作线程相同。

稍后会看到如何同时使用多个工作线程……

或者，还可以根据一个不同的JavaScript文件创建其他工作线程。

pin51.com 拼吾爱

编写Manager.js

既然你已经了解如何创建一个工作线程（而且也知道了这是多么容易），下面就来完成我们的manager.js代码。我们会力求代码简单，目前先创建一个工作线程。创建一个名为manager.js的文件，并增加以下代码：

```
window.onload = function() {
    var worker = new Worker("worker.js");
}
```

等待页面完全加载。

然后创建一个新的工作线程。

这是一个很好的起点，不过现在我们还希望这个工作线程具体做些工作。前面已经讨论过，要让一个工作线程做一些工作，一种方法是向它发送一个消息。要发送消息，需要使用工作线程对象的postMessage方法。用法如下：

```
window.onload = function() {
    var worker = new Worker("worker.js");

    worker.postMessage("ping");
}
```

使用工作线程的postMessage方法向它发送一个消息。我们的消息是一个简单的字符串"ping"。

postMessage方法在Web工作线程API中定义。

想发送更复杂的消息？可以这样做……

postMessage特写

postMessage中除了能发送字符串，还可以发送更复杂的消息。
下面来看消息中能发送哪些内容：

```
worker.postMessage("ping");
worker.postMessage([1, 2, 3, 5, 11]);
worker.postMessage({"message": "ping", "count": 5});
```

可以发送一个字符串……

……数组……

……甚至JSON对象。

但不能发送函数：

```
worker.postMessage(updateTheDOM);
```

不能发送函数……因为函数中可能包含一个DOM引用，导致工作线程有可能改变DOM！

拼吾爱

从工作线程接收消息

manager.js代码还远没有完成。如果我们想利用工作线程的劳动成果，还要能够从工作线程接收消息。要想接收一个工作线程的消息，需要为工作线程的onmessage属性定义一个处理程序，一旦这个工作线程发来消息，就会调用我们的处理程序（并传入消息）。可以这样做：

```
window.onload = function() {
    var worker = new Worker("worker.js");

    worker.postMessage("ping");

    worker.onmessage = function (event) {
        var message = "Worker says " + event.data;
        document.getElementById("output").innerHTML = message;
    };
}
```

在这里，我们定义了一个函数，只要从这个工作线程接收到一个消息就会调用这个函数。来自工作线程的消息包装在一个事件对象中。

传入处理程序的事件对象有一个data属性，其中包含工作线程提交的消息数据（这正是我们想要的）。

从工作线程得到一个消息时，把它放在HTML页面上的一个<p>元素中。

onMessage特写

下面来简单了解onmessage处理程序从工作线程接收到的消息。前面我们说过，这个消息包装在一个Event对象中，我们对它的两个属性感兴趣：data和target：

```
worker.onmessage = function (event) {
    var message = event.data;
    var worker = event.target;
};
```

这就是工作线程提交一个消息时从工作线程发送到页面代码的对象。

data属性包含工作线程发送的消息（例如一个字符串，如"pong"）。

target是发出这个消息的工作线程的一个引用。如果需要知道这个消息来自哪个工作线程，这就很方便。本章后面还会用到这个属性。

pin51.com 拼吾爱 最新最全资源分享下载站

现在来编写工作线程

要编写工作线程，首先需要确保这个工作线程可以接收从manager.js发出的消息，这样工作线程才能得到工作号令。为此，我们还要使用另一个onmessage处理程序，这是工作线程自身的一个处理程序。每个工作线程都可以接收消息，只需要为工作线程提供一个处理程序来处理接收到的消息。我们的做法如下（你可以创建一个文件worker.js，并增加以下代码）：

```
onmessage = pingPong;
```

将工作线程的onmessage属性赋为pingPong函数。

我们要编写函数pingPong来处理所有到来的消息。

编写工作线程的消息处理程序

下面来编写工作线程的消息处理程序pingPong，先从简单的开始。它的工作是这样的（你可能已经从pingPong这个名字猜到了）：工作线程先要检查它得到的消息，确保其中包含字符串“ping”，如果确实包含这个字符串，再发回一个消息“pong”。所以，实际上这个工作线程的工作就是得到一个“ping”，回答一个“pong”。这里不打算做任何繁重的计算工作，我们只希望确认管理器和工作线程之间能够正常通信。噢，对了，如果收到的消息中不包含“ping”，就将它忽略。

所以函数pingPong会得到一个消息，并响应“pong”。下面为worker.js增加以下代码：

```
onmessage = pingPong;
function pingPong(event) {
    if (event.data == "ping") {
        postMessage("pong");
    }
}
```

工作线程从主代码接收一个消息时，会调用pingPong函数，并传入这个消息。

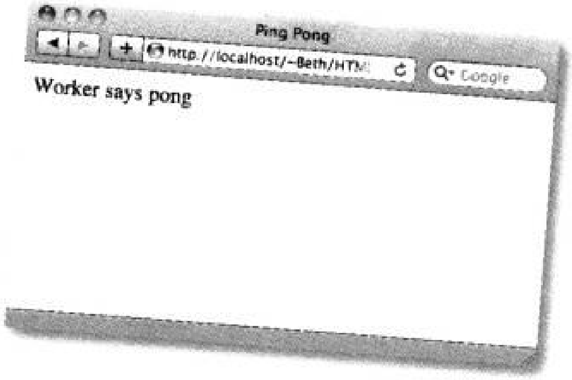
如果这个消息中包含字符串“ping”，就发回一个消息“pong”。工作线程的消息将返回给创建这个工作线程的代码。

注意工作线程也使用postMessage来发送消息。

pin51.com 拼吾爱 最新最全资源分享下载站

做个测试

确保你已经键入并保存了文件pingpong.html、manager.js和worker.js。现在打开这些文件，以便查看。下面来考虑具体是如何工作的。首先，manager.js创建一个新的工作线程，为它指定一个消息处理程序，然后向这个工作线程发送一个“ping”消息。接下来，工作线程将pingPong设置为它的消息处理程序，然后等待。在某个时刻，工作线程从管理器接收到一个消息，接收到消息时它会查看其中是否包含“ping”，如果确实包含，这个工作线程会做~~大量~~很少的工作，发回一个“pong”消息。



此时，主浏览器代码会从工作线程接收到一个消息，它将这个消息传递给消息处理程序。消息处理程序再在这个消息的前面追加“Worker says ”，并显示这个字符串。

根据这个执行过程，现在页面应当显示“Worker says pong”……好了，好了，我们知道了，你不用担心…… 加载这个页面吧！



pin51.com 拼吾爱

扮演浏览器



现在来扮演执行JavaScript的浏览器。对于下面的每段代码，把你当成一个浏览器，在这里给出的线上写出各个代码的输出。可以假设这个代码使用我们刚才编写的*worker.js*：

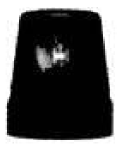
← 可以对照检查本章最后给出的答案。

```
window.onload = function() {
    var worker = new Worker("worker.js");
    worker.onmessage = function(event) {
        alert("Worker says " + event.data);
    }
    for (var i = 0; i < 5; i++) {
        worker.postMessage("ping");
    }
}
```

```
window.onload = function() {
    var worker = new Worker("worker.js");
    worker.ommessage = function(event) {
        alert("Worker says " + event.data);
    }
    for(var i = 5; i > 0; i--) {
        worker.postMessage("pong");
    }
}
```

pin5i.com 拼吾爱 最新最全资源分享下载站

```
window.onload = function() {
    var worker = new Worker("worker.js");
    worker.onmessage = function(event) {
        alert("Worker says " + event.data);
        worker.postMessage("ping");
    }
    worker.postMessage("ping");
}
```



尝试这两个代码时要当心，你可能必须关闭浏览器才能退出……

```
window.onload = function() {
    var worker = new Worker("worker.js");
    worker.onmessage = function(event) {
        alert("Worker says " + event.data);
    }

    setInterval(pinger, 1000);

    function pinger() {
        worker.postMessage("ping");
    }
}
```

pin51.com 拼吾爱

Sharpen your pencil

虽然工作线程一般通过消息得到工作号令，但并非必须如此。下面给出一种简洁的方法，可以让工作线程和HTML漂亮地完成工作。如果你看懂了这个代码要做什么，请在下面给出描述。可以对照检查本章最后给出的答案。

quote.html

```
<!doctype html>
<html lang="en">
    <head>
        <title>Quote</title>
        <meta charset="utf-8">
    </head>
<body>
    <p id="quote"></p>
    <script>
        var worker = new Worker("quote.js");
        worker.onmessage = function(event) {
            document.getElementById("quote").innerHTML = event.data;
        }
    </script>
</body>
</html>
```

quote.js

```
var quotes = ["I hope life isn't a joke, because I don't get it.",
              "There is a light at the end of every tunnel... just pray it's not a train!",
              "Do you believe in love at first sight or should I walk by again?"];
var index = Math.floor(Math.random() * quotes.length);
postMessage(quotes[index]);
```

你的描述写在这里。

输入代码并试着运行！

....

....

....

Exercise

下面为我们的pingPong游戏增加几个工作线程。你的任务是在下面填空，完成这个代码，从而向工作线程发出3个ping，并从工作线程发回3个pong。

```
window.onload = function() {
    var numWorkers = 3;
    var workers = [];
    for (var i = 0; i < ________; i++) {
        var worker = new ________("worker.js");
        worker.________ = function(event) {

            alert(event.target + " says "
                            + event.________);
        };
        workers.push(worker);
    }
    for (var i = 0; i < ________; i++) {
        workers[i].________("ping");
    }
}
```

我们要创建3个工作线程，把它们存储在一个数组workers中。

在空中填入你的代码。

这里将这个新的工作线程增加到workers数组。

there are no Dumb Questions

问： **创建工作线程时能不能直接传递一个函数，而不是利用一个JavaScript文件？这样看起来好像更容易，而且与JavaScript通常的做法更一致。**

答： 不，不能这么做。下面来告诉你为什么，要知道，工作线程的要求之一是它不能访问DOM（出于同样的原因，也不能访问主浏览器线程的任何状态）。如果可以向Worker构造函数传入一个函数，这个函数可能还包含DOM或主JavaScript代码的引用，这就会违反这个要求。所以，为了避免这个问题，Web工作线程的设计者选择的做法是只能传递一个JavaScript URL。

问： **在消息中向工作线程发送一个对象时，它会不会成为主页面和工作线程之间的一个共享对象？**

答： 不会的，发送一个对象时，工作线程会得到这个对象的一个副本。工作线程做出的任何修改都不会影响主页面中的对象。工作线程在另一个环境中运行，这与主页面所在的环境不同，所以你不能访问主页面所在环境中的对象。工作线程发出的对象也是如此，主页面只能得到工作线程所发送对象的一个副本。

问： **工作线程可以访问localStorage或做出XMLHttpRequest请求吗？**

答： 这是可以的，工作线程可以访问localStorage，也可以做出XMLHttpRequest请求。

pin51.com 拼吾爱 最新最全 下载站

Exercise Solution

下面为我们的pingpong游戏增加几个工作线程。你的任务是在下面填空，完成这个代码，从而向工作线程发出3个ping，并从工作线程发回3个pong。下面是我们的答案。

```
window.onload = function() {
    var numWorkers = 3;
    var workers = [];
    for (var i = 0; i < numWorkers; i++) {
        var worker = new Worker("worker.js");
        worker.onmessage = function(event) {

                alert(event.target + " says "
                                    + event.data);
        };
        workers.push(worker);
    }
    for (var i = 0; i < workers.length; i++) {
        workers[i].postMessage("ping");
    }
}
```

使用numWorkers迭代3次，创建3个工作线程（你完全可以改变这个变量，来增加更多工作线程）！

在主页面代码中建立消息处理程序，这里使用了工作线程的onmessage属性。

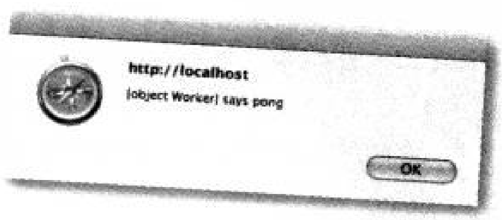
使用data属性来得到消息的内容。

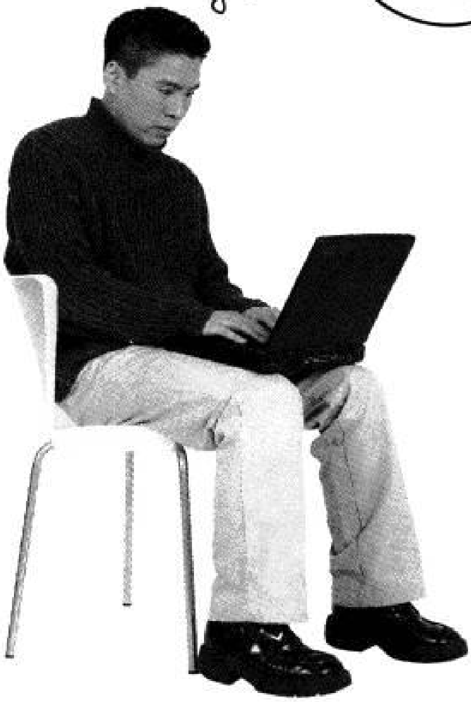
用postMessage向工作线程发出ping消息。

如果愿意，这里也可以使用numWorkers。

注意，不需要对工作线程代码做任何修改。每个工作线程都会独立地完成它的工作。

你会看到这个提醒出现3次。





可以看看importScripts。

Web工作线程有一个全局函数，名为`importScripts`，可以使用这个函数向工作线程中导入一个或多个JavaScript文件。使用`importScripts`时，要向它提供你想导入的文件或URL的一个列表（各个URL之间用逗号分隔），如下：

```
importScripts("http://bigscience.org/nuclear.js",
              "http://nasa.gov/rocket.js",
              "mylibs/atomsmasher.js");
```

importScripts中放入0个或多个用逗号分隔的JavaScript URL。

调用`importScripts`时，会按顺序获取和执行各个JavaScript URL。

注意`importScripts`是一个“货真价实”的函数，所以（不同于很多语言中的`import`语句），可以在运行时决定如何导入，如下：

```
if (taskType == "songdetection") {
   importScripts("audio.js");
}
```

因为importScripts是一个函数，所以可以按任务要求导入代码。

pin51.com 拼吾爱 最新最全的资源分享下载站

虚拟土地掠夺

Mandelbrot集的探险家们已经掠夺了这个虚拟空间的很多区域，给它们起了一些可爱的名字，比如说“海马谷”、“彩虹岛”，还有可怕的“黑洞”。从当前的实际房地产价值来看，似乎只有虚拟空间中的地产才能保值。所以，我们将为Mandelbrot集构建一个浏览器，加入这场虚拟土地的掠夺运动。实际上，必须承认，我们已经构建了一个Mandelbrot浏览器，不过它的速度很慢，在整个Mandelbrot集中导航可能要花很长时间，所以我们都希望能提高它的速度，要解决这个问题，我们有个预感：Web工作线程可能就是我们要找的答案。



像不像Azure Vortex边上的海滨地产？

四处看看

加载`http://wickedlysmart.com/hfhtml5/chapter10/singlethread/fractal.html`，你会看到Mandelbrot集的远距离图像。随意点击某个位置，就会放大这张图的一个区域。可以不断点击探索不同的区域，或者重新加载来重新开始。要当心有黑洞的区域，你可能会被吸进去再也出不来。我们不了解你的情况，不过，既然景色这么美，我们的浏览器最好能再快一点……你是不是也这么想？如果性能足够好，能够把视图最大化，占满整个浏览器窗口就太好了！下面将向Fractal Explorer增加Web工作线程来修正所有这些问题。

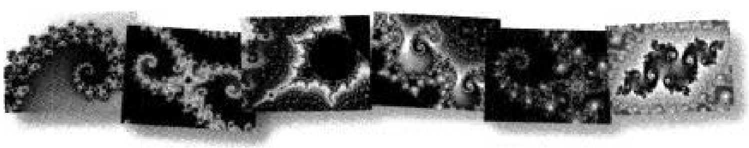


嗯，如果你正好是个数学家，可能知道Mandelbrot集就是下面这个公式：

$$z_{n+1} = z_n^2 + c$$

Mandelbrot集由Benoit Mandelbrot[译注]发现，并做了深入的研究。你可能还知道，Mandelbrot集就是由这个公式生成的一组复数（包含一个实部和一个虚部的数）。

另一方面，如果你不是一个数学家，考虑Mandelbrot集最好的方式就是把它看作是一个无限复杂的分形图像。这表示这个图像可以无限放大，放大到能想象到的任何程度，可以从中找到一些有趣的结构。下面就是在这个Mandelbrot集中导航找到的一些图像：



为什么我们对这个Mandelbrot集这么感兴趣？嗯，这是因为Mandelbrot集有很多有趣的特性。首先，它是由一个非常简单的公式生成的（就是上面那个公式），这个公式只需要几行代码就可以表示。其次，生成Mandelbrot集需要大量计算周期，所以可以非常适合作为使用Web工作线程的一个例子。最后一点，嘿，这个应用很酷，实现起来很有意思，把它作为这本书的终极应用非常合适，你不这么认为吗？

愿他安息，Benoit Mandelbrot，写这本书期间他刚刚过世。认识你真是我们的荣幸。

译注：Benoit Mandelbrot，波兰出生的法国数学家，作为分形几何学的创始人闻名于世。

pin51.com

如何计算Mandelbrot集

在用到工作线程之前，下面先来看通常如何构建代码来计算一个Mandelbrot集。我们并不想过多地关注计算Mandelbrot像素值的具体细节；在这方面，我们已经得到了负责完成这些工作的全部代码，稍后就会提供给你。对现在来说，我们只希望你对于如何计算Mandelbrot集有一个全局认识：

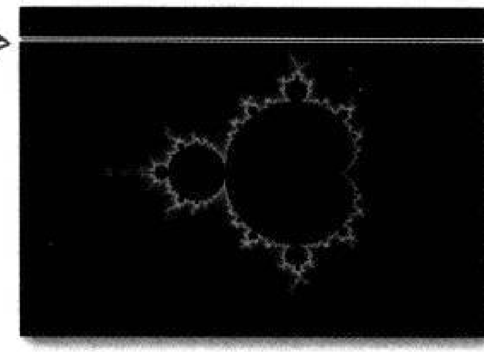
需要说明，我们的目标并不是教你成为一个数值分析学家（会写复数公式），真正的目标是让你修改这个计算密集的应用，来使用Web工作线程。如果你对Mandelbrot集的数学方面感兴趣，可以参考Wikipedia来了解，这是一个不错的起点。

要计算Mandelbrot集，需要循环处理图像的每一行。

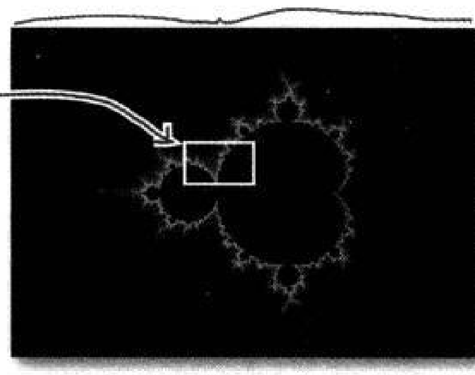
```
for (i = 0; i < numberOfRows; i++) {
    var row = computeRow(i);
    drawRow(row);
}
```

对于每一行，需要计算这一行的像素。

然后在屏幕上绘制每一行。在你的浏览器上运行这个测试代码时，可能会看到图像会逐行显示。



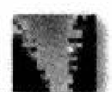
现在这个代码只是简单的伪代码，编写真正的实际代码时，还有一些细节问题需要考虑。例如，计算一行时，我们需要知道这一行的宽度和放大因子，还要知道按怎样的分辨率（精度）来计算，以及另外一些小细节。可以把所有这些细节放在一个任务对象中，如下：




```
for (i = 0; i < numberOfRows; i++) {
    var taskForRow = createTaskForRow(i);
    var row = computeRow(taskForRow);
    drawRow(row);
}
```

taskForRow对象包含计算一行所需的全部数据。

将taskForRow传入computeRow，它会返回计算得到的行。



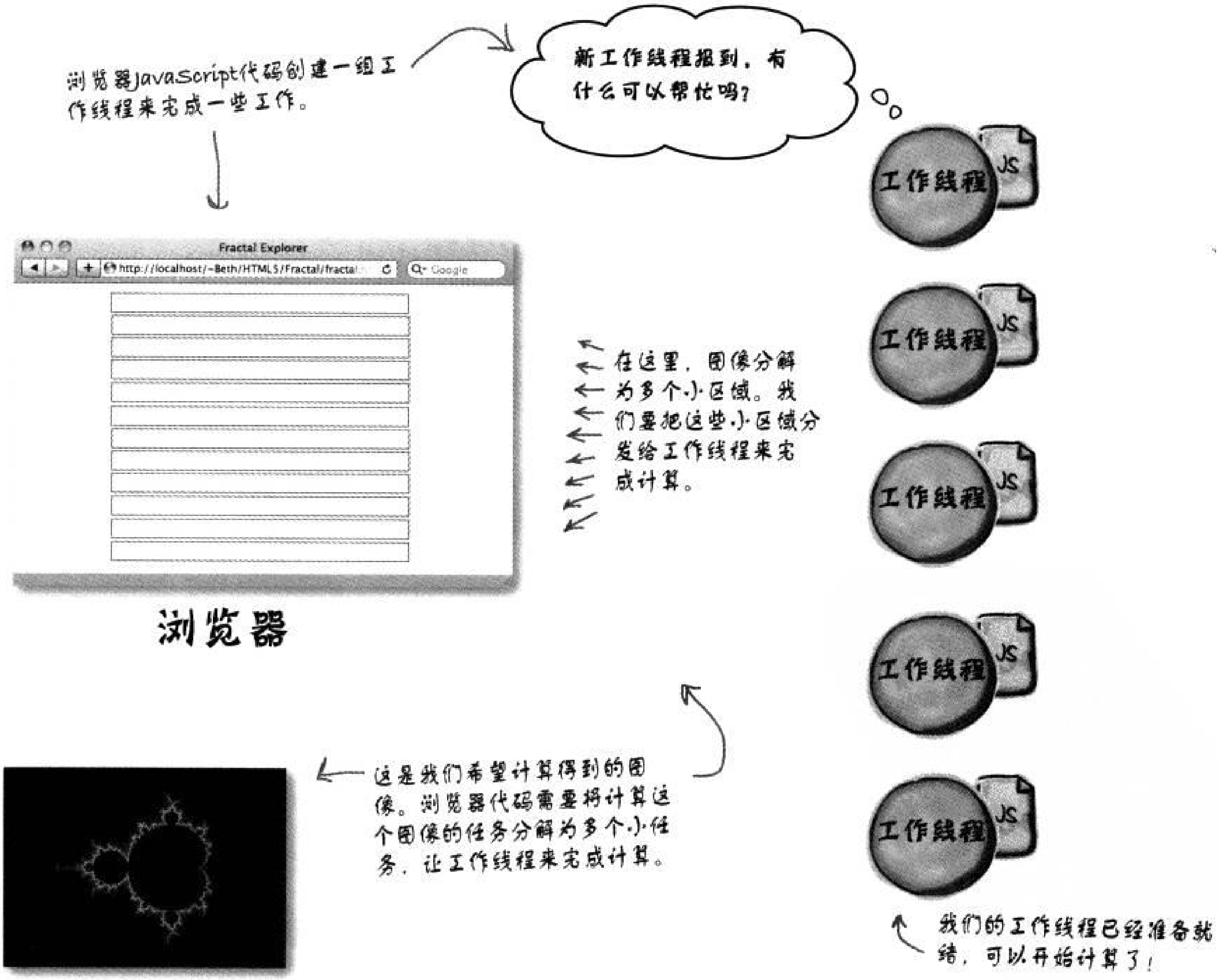
现在的难题是需要修改这个代码，将计算分解到多个工作线程，然后增加代码向工作线程分发任务，并处理这些工作线程完成任务时提交的结果。

www.pin51.com 拼吾爱 最新最全的电子书下载站

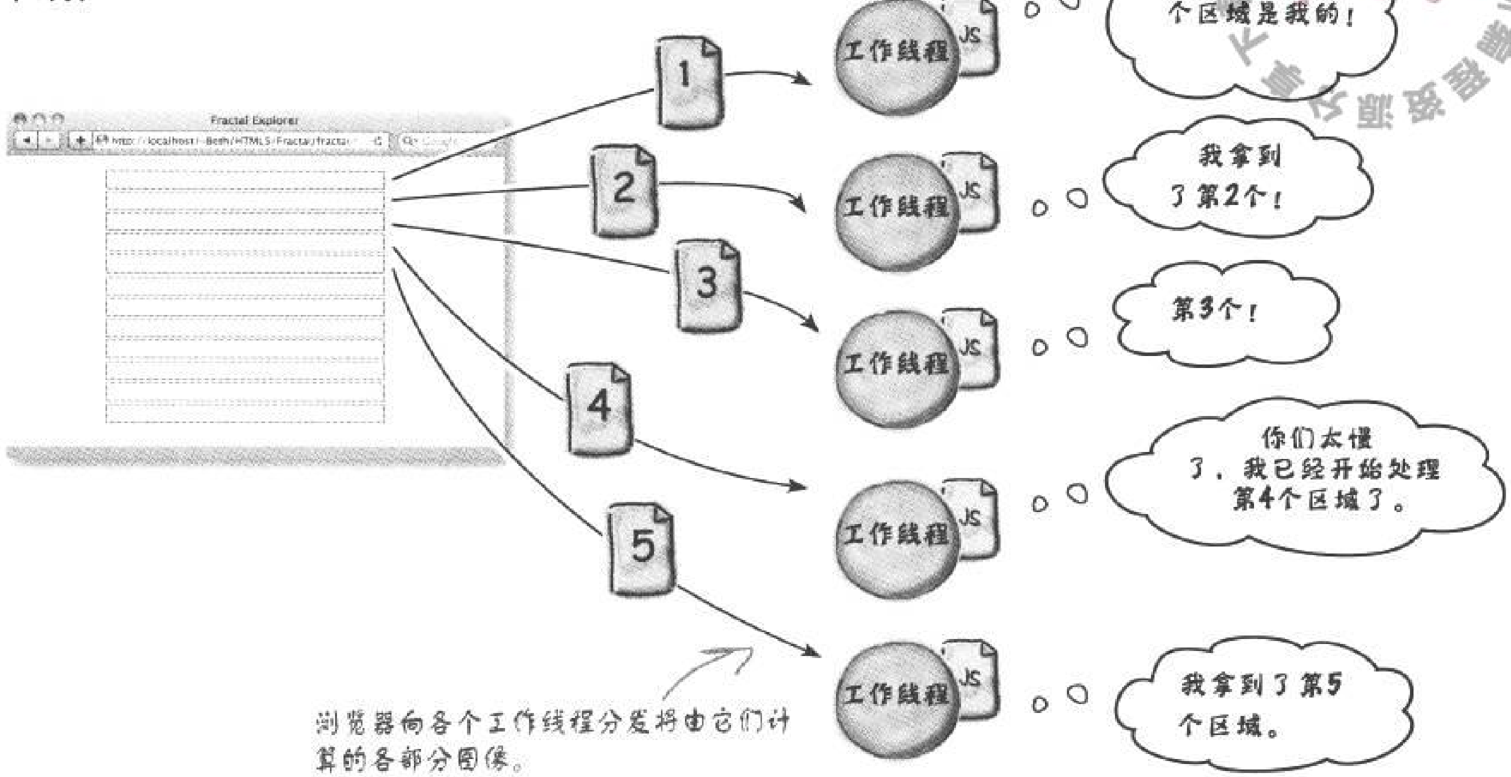
如何使用多个工作线程

你已经知道如何创建新的工作线程，不过怎么使用这些工作线程来完成更复杂的工作呢？比如计算Mandelbrot集的行？或者向一个图像应用一种类Photoshop的特效？再或者对一个电影场景完成光线跟踪？对于所有这些情况，都可以把工作分解为小任务，让各个工作线程独立地完成。现在我们还是来计算Mandelbrot集（不过这里使用的模式完全可以应用于以上任何例子）。

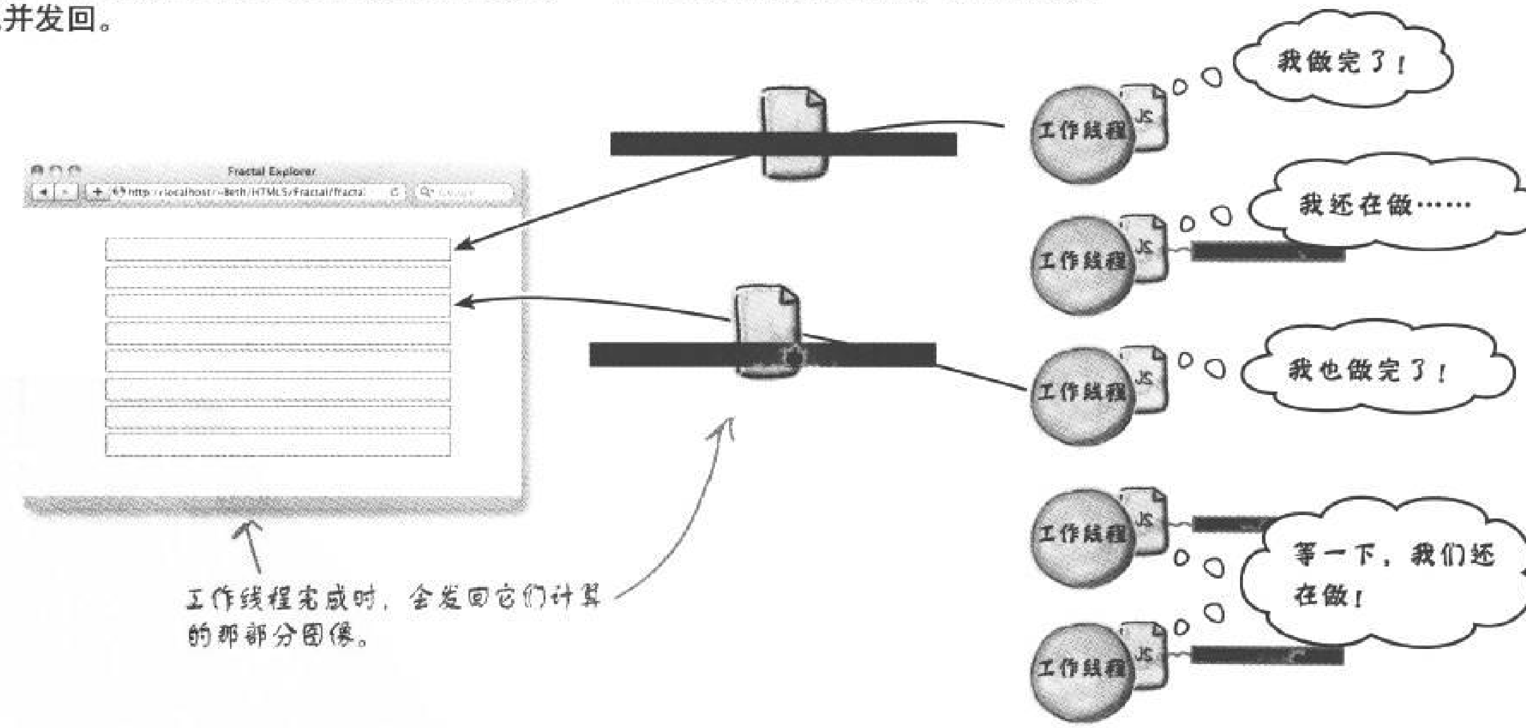
开始时，浏览器首先创建一组工作线程来帮忙（不过不能太多，如果创建太多的工作线程，代价会很昂贵，后面还会介绍更多有关内容）。这个例子中只使用5个工作线程：



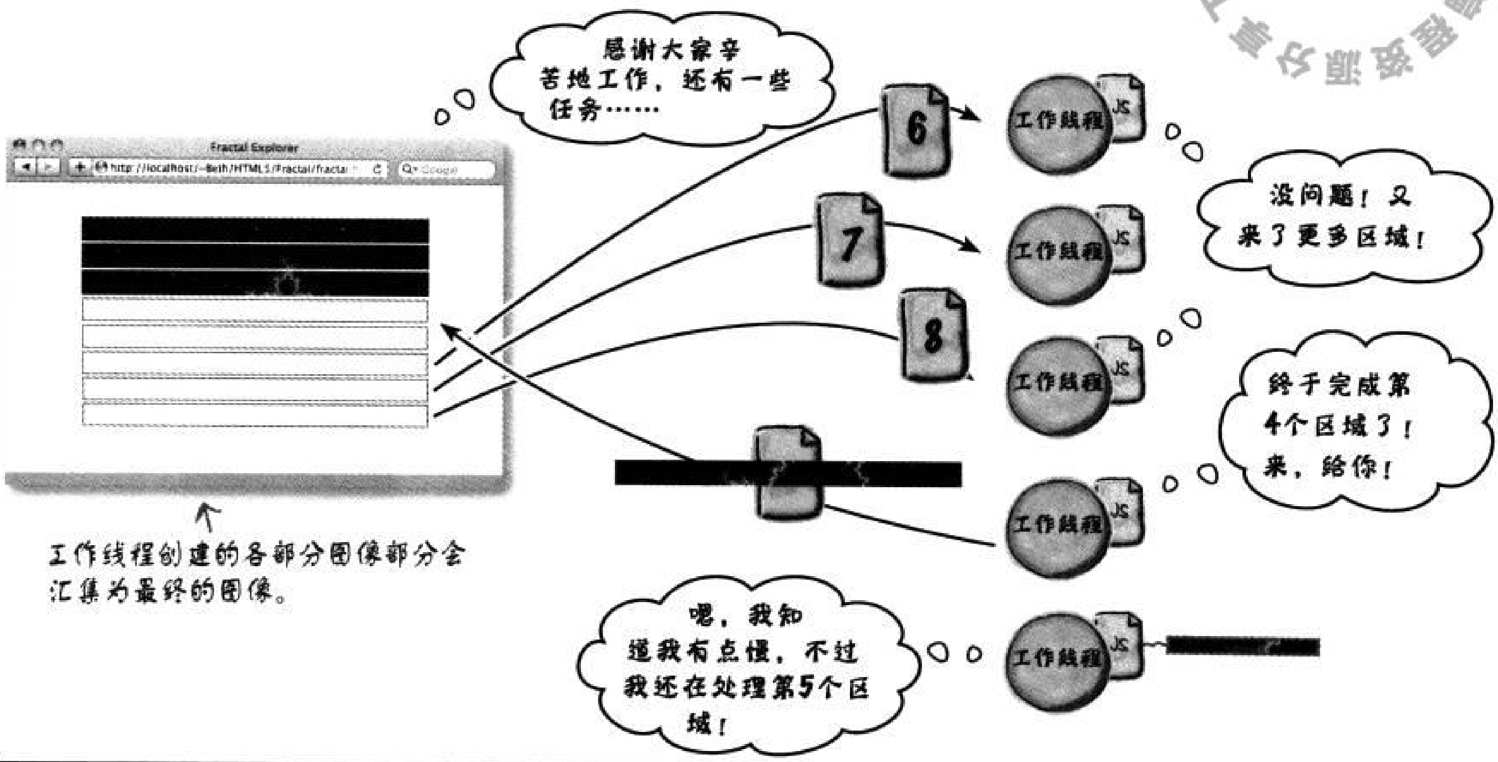
接下来，浏览器代码将图像的不同部分分发给各个工作线程来计算：



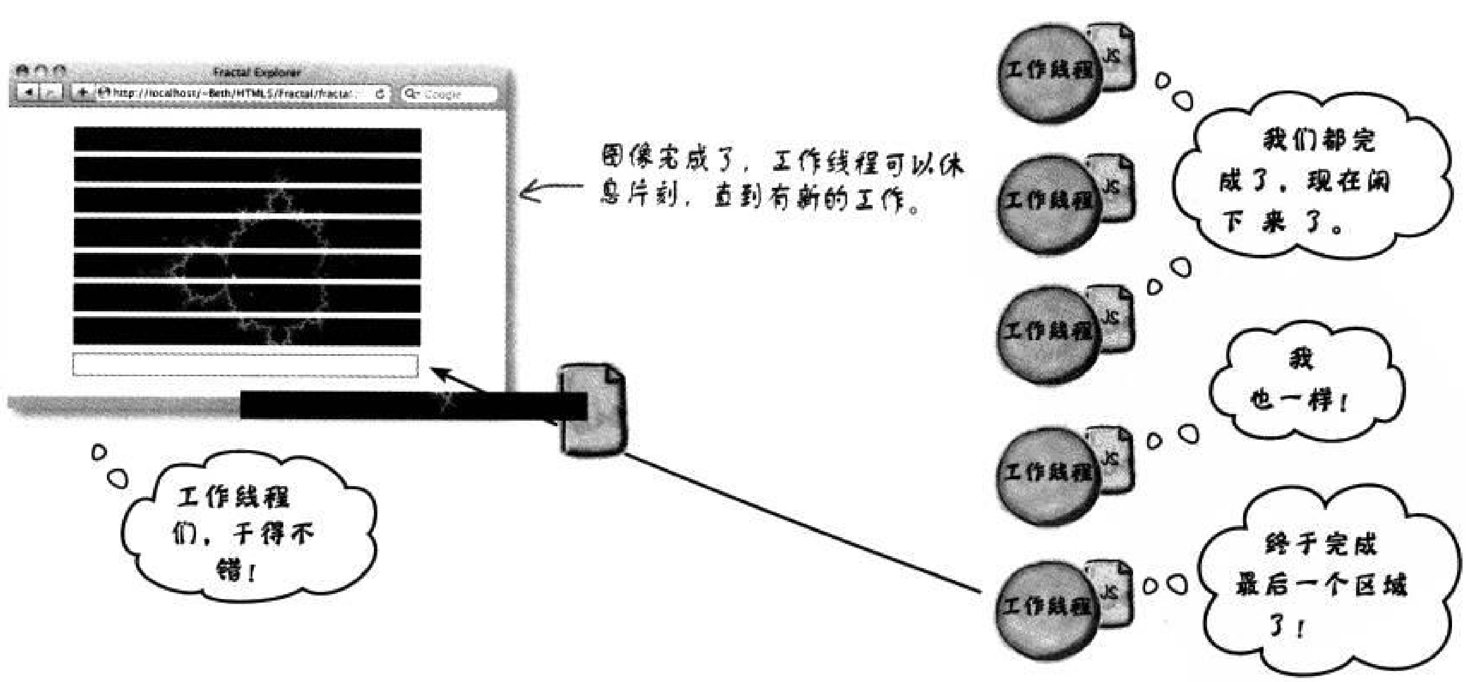
每个工作线程独立地处理它的那部分图像。一个工作线程完成任务时，将把结果打包并发回。



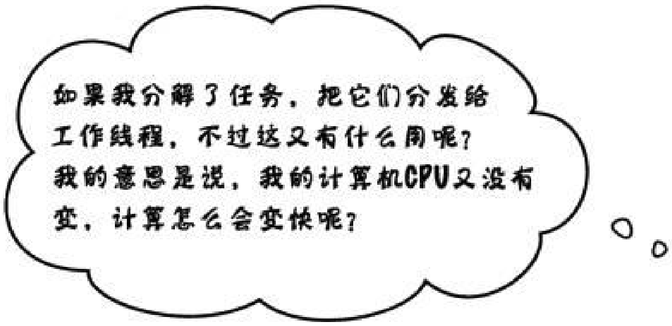
从工作线程返回各部分图像时，它们将汇集到浏览器中的图像，如果还有更多部分需要计算，会向空闲的工作线程分发新任务。



计算最后一部分图像后，图像就完成了，工作线程将空闲，直到用户点击放大，然后一切又开始了……



pin51.com 拼吾爱



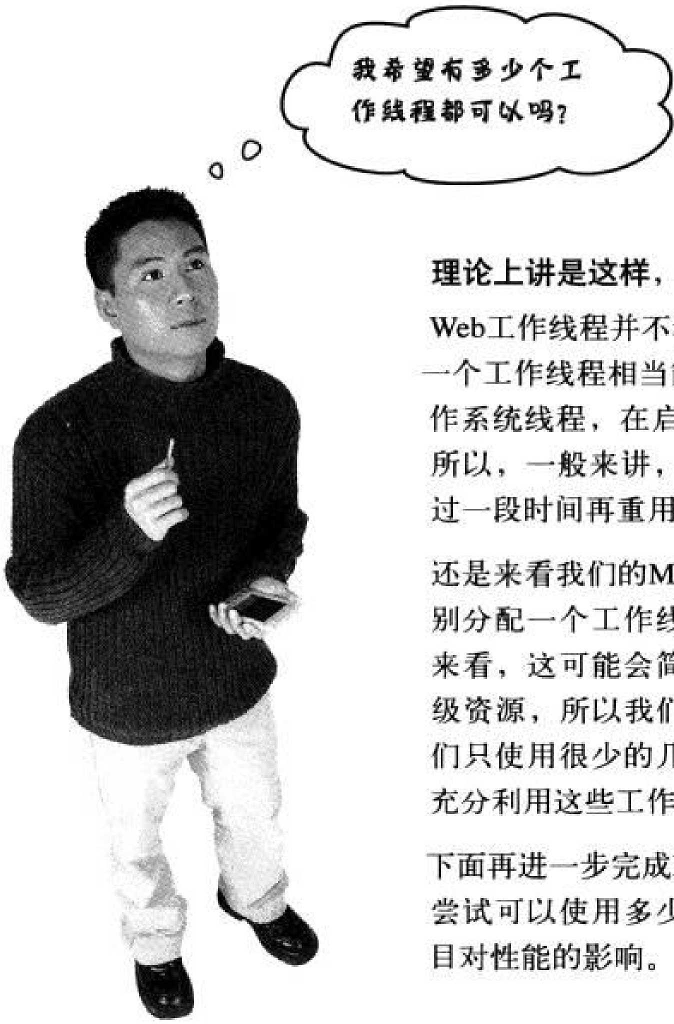


要想变快有两种方法……

首先考虑一个应用需要做大量“计算”，同时它还必须对用户做出响应。如果这个应用占用了大量JavaScript时间，用户就会看到界面很迟缓，感觉很慢（重申一次，这是因为JavaScript是单线程的）。向这样一个应用增加工作线程，就能立即改善用户的应用体验。为什么？这是因为JavaScript在从工作线程得到结果的间隙将有机会对用户交互做出响应，倘若一切都在主线程中计算，它是根本没有这样的机会的。所以用户界面会更有响应性，你的应用也会感觉更快（尽管底层运行并没有加快）。不相信吗？你可以试试看，请一些真正的用户使用你的应用。问问他们的感受。

第二种方法确实会更快。如今几乎所有现代桌面计算机和设备都配备有多核处理器（可能甚至还有多个处理器）。多核就意味着处理器可以并发地完成多个工作。如果只有一个控制线程，浏览器中JavaScript就无法利用你的多核或多处理器，它们只能被浪费。不过，如果使用Web工作线程，工作线程将充分利用多核，可以在不同的核上运行，所以你会看到应用速度的真正提升，因为现在利用了更多的处理器能力。如果你有一个多核的机器，拭目以待吧，很快你就会看到明显差别。

pin51.com 拼吾爱 最新资源分享下载站



理论上讲是这样，但实践中不行。

Web工作线程并不希望大量使用。尽管从代码来看创建一个工作线程相当简单，但这需要额外的内存和一个操作系统线程，在启动时间和资源方面开销可能很昂贵。所以，一般来讲，你可能只希望创建有限的工作线程，过一段时间再重用这些工作线程。

还是来看我们的Mandelbrot例子。理论上讲，你可以分别分配一个工作线程来计算每一个像素，从代码设计来看，这可能会简单得多，但是由于工作线程是重量级资源，所以我们绝对不会那样设计应用。相反，我们只使用很少的几个工作线程，并适当地设计计算来充分利用这些工作线程。

下面再进一步完成Fractal Explorer的设计，然后再回来尝试可以使用多少个工作线程，从而理解工作线程数目对性能的影响。

pin51.com 拼吾爱 最新最全 下载站

BRAIN POWER

你现在已经掌握了不少基础知识，知道如何构建Web工作线程应用，也了解如何创建和使用工作线程，另外对于任务分解也有所了解，知道如何将大规模计算分解为小任务，可以由工作线程分别计算，从而解决这些大规模的计算问题，甚至对如何计算Mandelbrot集也略知一二。试着把这些知识综合起来，考虑如何重写下面的伪代码来使用工作线程。可以先假设需要多少工作线程就有多少（假设对应每一行都有一个工作线程），然后再增加约束，要求有限数目的工作线程（工作线程少于行数）：

```
for (i = 0; i < numberOfRows; i++) {
    var taskForRow = createTaskForRow(i);
    var row = computeRow(taskForRow);
    drawRow(row);
}
```

这是我们现在的伪代码，要增加Web工作线程需要怎么做？

你的说明写在这里：

pin51.com 拼吾爱

构建Fractal Explorer应用

- [ ] 创建HTML
- [ ] 成品代码
- [ ] 创建工作线程
- [ ] 启动工作线程
- [ ] 实现工作线程
- [ ] 处理结果
- [ ] 用户交互代码

我们需要做到：

- [ ] 建立HTML页面包含Mandelbrot应用。
- [ ] 输入（或下载）成品代码。
- [ ] 创建一些工作线程来完成计算。
- [ ] 启动工作线程完成任务。
- [ ] 实现工作线程代码。
- [ ] 工作线程完成任务时处理工作线程的结果。
- [ ] 在用户界面中处理点击和大小调整事件。

创建分形浏览器HTML标记

首先需要建立一个HTML页面包含我们的应用。可以创建一个名为`fractal.html`的HTML文件，并增加以下标记。下面来分析这个文件：

```
<!doctype html>
<html lang="en">
  <head>
    <title>Fractal Explorer</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="fractal.css">
    <script src="mandellib.js"></script>
    <script src="mandel.js"></script>
  </head>
  <body>
    <canvas id="fractal" width="800" height="600"></canvas>
  </body>
</html>
```

与以往一样，这是一个标准的HTML5文件。

这是我们为你提供的成品代码全部，其中包含所有数值计算代码，还有处理图形的一些代码。

这里是我们要编写的JavaScript代码……

如果你不知道工作线程代码怎么用，要记住，我们不会直接链接到一个工作线程JavaScript文件，而是会在创建工作线程时引用这个文件。

看！我们的老朋友<canvas>！

<body>有一个canvas元素。将它的初始大小设置为800×600像素，不过后面会看到如何使用JavaScript把它的大小调整为窗口的宽度和高度。毕竟，我们希望Mandelbrot集尽可能大！

这个代码放在fractal.html中。

min51.com 拼吾爱



成品代码

提醒：可以从http://wickedlysmart.com/hfhtml5下载所有代码。

必须告诉你，我们原本计划用整个一章来介绍计算Mandelbrot集的所有趣闻轶事……我们打算给出详细解释，包括Benoit Mandelbrot的经历，他如何发现Mandelbrot集，Mandelbrot集的所有惊人特性、像素优化、色图等，不过后来编辑打来电话，你知道的，这可是很重要的电话。我想我们的进度可能晚了一点，所以赶紧致歉，调整了计划，不过还是需要提供一些成品代码来完成Mandelbrot图形的底层计算。但这样一来也有好的一方面，我们可以专注于如何使用Web工作线程，而不用再花好几天时间研究数学和图像问题。不过，我们还是建议你自己好好研究这些主题！

不管怎样，首先要有管理任务的代码，还需要一些代码来绘制分形图像中的各行。先把这个代码键入到一个名为“mandellib.js”的文件中：

```
var canvas;
var ctx;

var i_max = 1.5;
var i_min = -1.5;
var r_min = -2.5;
var r_max = 1.5;

var max_iter = 1024;
var escape = 1025;
var palette = [];

function createTask(row) {
    var task = {
        row: row,
        width: rowData.width,
        generation: generation,
        r_min: r_min,
        r_max: r_max,
        i: i_max + (i_min - i_max) * row / canvas.height,
        max_iter: max_iter,
        escape: escape
    };
    return task;
}
```

注意这里是我们熟悉的画布和上下文。

这些是Mandelbrot图形代码用来计算和显示Mandelbrot集的全局变量。

这个函数将工作线程计算一行像素所需的全部数据打包为一个对象。后面你会看到如何将这个对象传给工作线程加以使用。

这个代码放在mandellib.js文件中。

www.minxue51.com 最新资源 分享吾爱

成品代码（续）……

- [x] 创建HTML
- [ ] 成品代码
- [ ] 创建工作线程
- [ ] 启动工作线程
- [ ] 实现工作线程
- [ ] 处理结果
- [ ] 用户交互代码

```
function makePalette() {
    function wrap(x) {
        x = ((x + 256) & 0x1ff) - 256;
        if (x < 0) x = -x;
        return x;
    }
    for (i = 0; i <= this.max_iter; i++) {
        palette.push([wrap(7*i), wrap(5*i), wrap(11*i)]);
    }
}
```

makePalette将一个庞大的数字集映射到一个RGB颜色数组。我们将在（下面的）drawRow中使用这个调色板，将从工作线程得到的值转换为一个颜色，来完成分形集的图形显示（分形图像）。

drawRow从工作线程得到结果，把它们绘制到画布上。

```
function drawRow(workerResults) {
    var values = workerResults.values;
    var pixelData = rowData.data;
    for (var i = 0; i < rowData.width; i++) {
        var red = i * 4;
        var green = i * 4 + 1;
        var blue = i * 4 + 2;
        var alpha = i * 4 + 3;
        pixelData[alpha] = 255; // set alpha to opaque
        if (values[i] < 0) {
            pixelData[red] = pixelData[green] = pixelData[blue] = 0;
        } else {
            var color = this.palette[values[i]];
            pixelData[red] = color[0];
            pixelData[green] = color[1];
            pixelData[blue] = color[2];
        }
    }
    ctx.putImageData(this.rowData, 0, workerResults.row);
}
```

使用这个rowData变量来完成工作。rowData是一个单行ImageData对象，包含画布中这一行的实际像素。

这里使用调色板将从工作线程得到的结果（一个数）映射为一个颜色。

这个代码应该很熟悉了。它类似于我们在第8章对视频和画布所做的工作。

这里将像素写入画布上下文中的ImageData对象！

这个代码放在mandellib.js文件中。

pin51.com 拼吾爱 最新最全资源下载站

成品代码（续）……

setUpGraphics会建立所有图形绘制代码以及Mandelbrot计算所使用的全局变量。

```
function setupGraphics() {

    canvas = document.getElementById("fractal");
    ctx = canvas.getContext("2d");

    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;

    var width = ((i_max - i_min) * canvas.width / canvas.height);
    var r_mid = (r_max + r_min) / 2;
    r_min = r_mid - width/2;
    r_max = r_mid + width/2;

    rowData = ctx.createImageData(canvas.width, 1);

    makePalette();
}
```

这里获取画布和上下文，并设置画布的初始宽度和高度。

这些是用来计算Mandelbrot集的变量。

这里初始化rowData变量（用来向画布写像素）。

这里初始化调色板，用来将Mandelbrot集绘制为一个分形图像。

这个代码放在mandellib.js文件中。

pin5i.com 拼吾爱 最新最全下载站

成品代码（续）……

下面这个成品代码就是工作线程用来完成Mandelbrot集数学计算的具体代码。Mandelbrot集计算的魔力就在这里体现（如果你想更深入地研究Mandelbrot集，这也是你要关注的重点）。把这个代码键入到“workerlib.js”：

computeRow计算Mandelbrot集的一行数据。要为这个函数提供一个对象，计算一行数据所需的值都打包在这个对象中。

这里有大量计算。太好了！

注意，对于要显示的每一行，我们要完成两个循环，一个对应行中的各个像素……

```
function computeRow(task) {
    var iter = 0;
    var c_i = task.i;
    var max_iter = task.max_iter;
    var escape = task.escape * task.escape;
    task.values = [];
    for (var i = 0; i < task.width; i++) {
        var c_r = task.r_min + (task.r_max - task.r_min) * i / task.width;
        var z_r = 0, z_i = 0;

        for (iter = 0; z_r*z_r + z_i*z_i < escape && iter < max_iter; iter++) {
            // z -> z^2 + c
            var tmp = z_r*z_r - z_i*z_i + c_r;
            z_i = 2 * z_r * z_i + c_i;
            z_r = tmp;
        }
        if (iter == max_iter) {
            iter = -1;
        }
        task.values.push(iter);
    }
    return task;
}
```

……另一个循环用来查找对应这个像素的适当的值。这个内循环决定了计算复杂性，也正是因为有这个内循环，在多核计算机上运行这个代码会快得多！

所有这些计算的结果是得到一个值，这个值将增加到一个命名值数组中，这个数组再放入task对象，使得工作线程可以把结果发回给主代码。

稍后会更详细地分析这一部分。

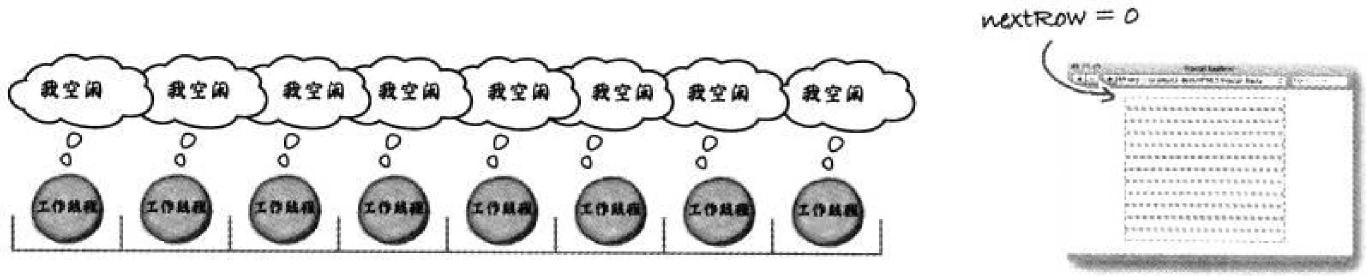
这个代码放在workerlib.js文件中。



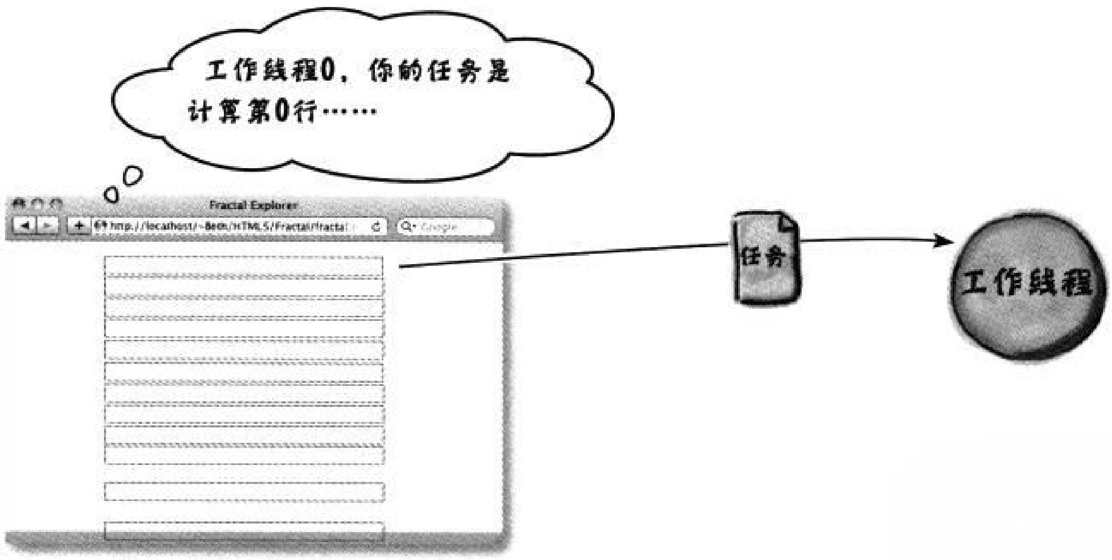
创建工作线程，为它们分配任务……

有了这些**成品代码**，下面再把重心转向编写具体代码，来创建工作线程并为工作线程分配任务。我们要这样做：

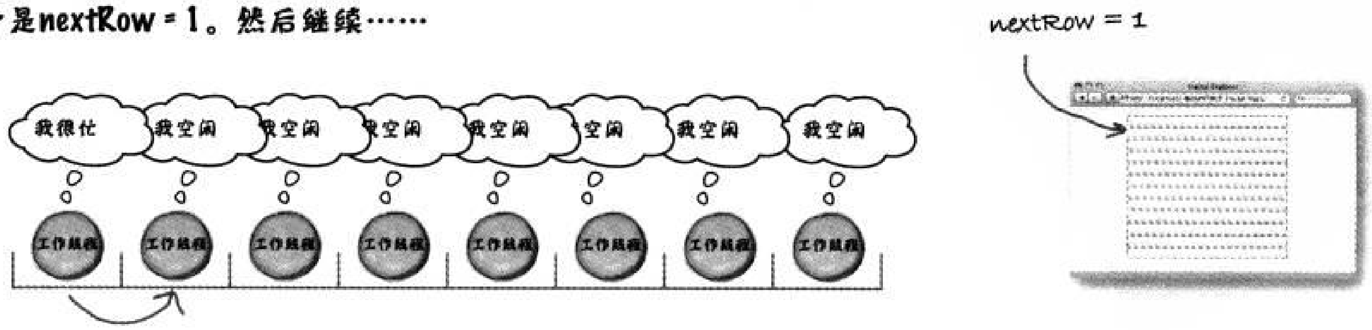
1. **创建一个工作线程数组，开始时这些工作线程都是空闲的。另外还有一个未计算任何内容的图像(nextRow = 0)。**



2. **迭代处理这个数组，为每个空闲工作线程创建一个任务：**



3. **继续迭代处理，查找下一个空闲的工作线程，为它分配任务。下一个是nextRow = 1。然后继续……**



- [x] 创建HTML
- [x] 编写成品代码
- [ ] 创建工作线程
- [ ] 启动工作线程
- [ ] 实现工作线程
- [ ] 处理结果
- [ ] 用户交互代码

编写代码

既然已经知道如何创建和管理工作线程，下面就来编写代码。为此需要一个初始函数，所以先在mandel.js中创建一个名为init的函数，还要在这个函数中加入另外一些代码来建立和运行应用（比如确保已经完成了图形初始化）：

首先，定义一个变量，包含我们希望的工作线程个数。我们选择工作线程数为8，完成这个应用后还可以尝试其他选择。

为什么是8？嗯，我们刚好有一个8核的计算机，所以这与我们的计算机能力恰好匹配。不过即使你没有8个核，作为尝试，8也是个不错的选择。

```
var numberOfWorkers = 8;
var workers = [];

window.onload = init;

function init() {
    setupGraphics();

    for (var i = 0; i < numberOfWorkers; i++) {
        var worker = new Worker("worker.js");

        worker.onmessage = function(event) {
            processWork(event.target, event.data);
        }

        worker.idle = true;

        workers.push(worker);
    }

    startWorkers();
}
```

这里是一个空数组，用来存放我们的工作线程。

下面建立一个onload处理程序，页面完全加载时会调用init。

这个函数在成品代码中定义，它会处理很多工作，包括得到画布上下文，将画布大小调整为浏览器的大小，还会处理另外一些图形方面的细节问题。

现在，根据工作线程的个数迭代处理……

……从"worker.js"创建一个新的工作线程，这个文件我们还没有编写。

然后将各个工作线程的消息处理程序设置为一个函数，它会调用processWork函数，将event.target（刚完成工作的工作线程）传入这个函数。

还有一点……要记住，我们希望知道哪些工作线程在工作，另外哪些是空闲的。为此，要向工作线程增加一个"idle"属性。这是我们自己定义的属性，而不属于Web工作线程API的一部分。目前先把它设置为true，因为我们还没有向工作线程分配任何工作。

将刚创建的工作线程增加到workers数组。

最后，需要在某个时刻让这些工作线程开始工作。将这些代码放在一个名为startWorkers的函数中，这个函数需要我们来编写。

这个代码放在mandel.js文件中。

启动工作线程

- [x] 创建HTML
- [x] 编写代码
- [x] 创建工作线程
- [ ] 启动工作线程
- [ ] 实现工作线程
- [ ] 处理结果
- [ ] 用户交互代码

OK，我们需要解决几个问题。首先要启动工作线程，还要编写能够处理工作线程返回结果的函数，另外，我们还需要为工作线程编写代码。下面首先编写启动工作线程的代码：

再向mandel.js增加两个全局变量。

```
var nextRow = 0;
var generation = 0;

function startWorkers() {
    generation++;
    nextRow = 0;

    for (var i = 0; i < workers.length; i++) {
        var worker = workers[i];

        if (worker.idle) {

            var task = createTask(nextRow);

            worker.idle = false;
            worker.postMessage(task);


            nextRow++;
        }
    }
}
```

第一个是nextRow，它会跟踪处理图像过程中处理到哪一行。

下一次用户放大Mandelbrot图像时，我们会开始计算一个新的图像。generation变量将跟踪计算了多少次。后面会对这个内容做更多介绍。

startWorkers函数将启动工作线程，如果用户放大图像，也会重启工作线程。所以，每次启动工作线程时，都会将nextRow重置为0，并让generation递增。

稍后就会更清楚地了解如何使用这两个变量……

现在循环处理workers数组中的所有工作线程……

……查看工作线程是否空闲。

如果空闲，就为这个工作线程分配一个任务。这个任务就是计算Mandelbrot集的一行。createTask在mandellib.js中定义，它会返回一个task对象，其中包含工作线程计算这一行所需的全部数据。

现在已经为工作线程分配了一些工作，所以将idle属性设置为false（这说明它现在很忙）。

在这里告诉工作线程开始工作，向它发出一个消息，其中包含这个任务。工作线程一直在监听消息，所以当它接收到这个消息时，就会开始处理这个任务。

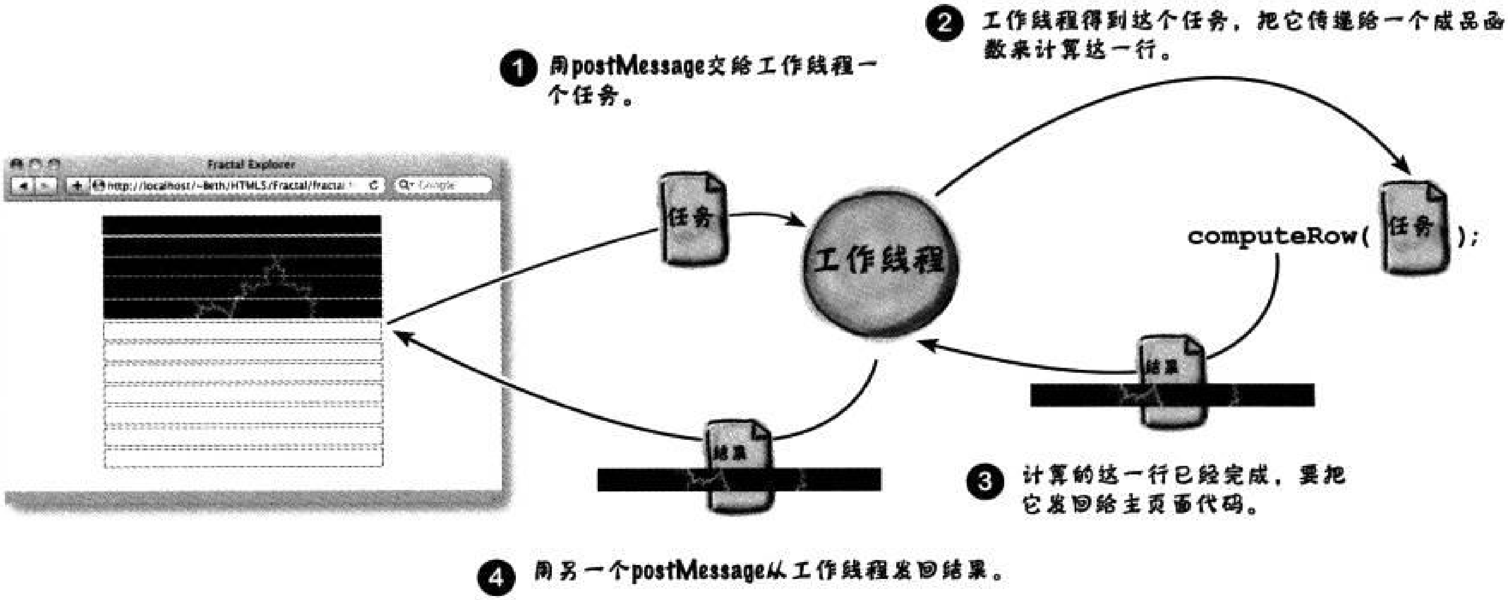
最后，将行数递增，所以下一个工作线程会得到下一行。

这个代码放在mandel.js文件中。

- [x] 创建HTML
- [x] 获取代码
- [x] 创建工作线程
- [x] 启动工作线程
- [ ] 实现工作线程
- [ ] 处理结果
- [ ] 用户交互代码

实现工作线程

既然已经有了启动工作线程的代码，还能为这些工作线程分别传入一个任务，下面就来编写工作线程代码。接下来，一旦工作线程计算完它那部分分形图像，我们所要做的就是处理来自工作线程的结果。不过，在为工作线程编写代码之前，先来简单回顾一下它的工作：



所以下面就来实现这个过程，可以把下面的代码键入到worker.js文件中。

使用importScripts导入workerlib.js成品代码，这样工作线程就可以调用这个库文件中定义的computeRow函数。

工作线程只需建立onmessage处理程序。它不需要做任何其他事情，因为它要做的就是等待来自mandel.js的消息开始工作！

```
importScripts("workerlib.js");

onmessage = function (task) {

    var workerResult = computeRow(task.data);

    postMessage(workerResult);

}
```

从task得到数据，并把这个数据传递到computeRow函数，这个函数会完成Mandelbrot计算的具体工作。

使用postMessage将计算的结果（保存在workerResult变量中）传回主JavaScript代码。

这个代码放在worker.js文件中。

pin51.com 拼吾爱 最新

稍事休息……

虽然只有几页，不过我们给出的代码可不少。下面稍稍休息一下，加加油，同时也填填肚子。

另外我们认为你可能想简单看看后台，了解一下工作线程的任务和结果究竟是什么样（后面会看到，它们非常相似）。所以，拿瓶汽水，趁休息我们来看一看……



任务特写

你已经见过createTask和postMessage调用，它们都使用了任务：

```
var task = createTask(nextRow);
worker.postMessage(task);
```

你可能想知道任务（task）究竟是什么样。嗯，它是一个由属性和值构成的对象，下面就来看一看：

```
task = {
    row: 1,
    width: 1024,
    generation: 1,
    r_min: 2.074,
    r_max: -3.074,
    i: -0.252336,
    max_iter: 1024,
    escape: 1025
};
```

任务包含工作线程完成计算所需的全部值。

row：标识要为哪一行创建像素值。

width：标识这一行的宽度。

generation：标识放大了多少次。后面会了解这个值如何使用……

r_min、r_max、i：这些定义了我们要计算的Mandelbrot区域。

max_iter、escape：这些控制了要计算的结果精度。

pin51.com 拼吾爱 最新最全资源分享下载站

结果特写

工作线程中计算行得到的结果呢？

```
var workerResult = computeRow(task.data);
postMessage(workerResult);
```

结果会是什么样？它们与任务非常相似：

```
workerResult = {
    row: 1,
    width: 1024,
    generation: 1,
    r_min: 2.074,
    r_max: -3.074,
    i: -0.252336,
    max_iter: 1024,
    escape: 1025,
    values: [3, 9, 56, ... -1, 22]
};
```

这些都与任务中一样。这一点很好，因为这样一来，从工作线程得到结果时，就能知道有关任务的一切情况。

工作线程得到传给它的任务，然后增加一个values属性，其中包含在画布上绘制这一行所需的数据。

哈，但这一个是新的。这些是各个像素的值，还需要映射到颜色（这在drawRow中完成）。



该上路了……

感谢你花时间陪我们查看了任务和结果。最好再喝口汽水，我们又该上路了！



www.lin51.com 无忧书

- [x] 创建HTML
- [x] 创建产品代码
- [x] 创建工作线程
- [x] 启动工作线程
- [x] 实现工作线程
- [ ] 处理结果
- [ ] 用户交互代码

重回代码：如何处理工作线程的结果

既然已经了解工作线程的结果，下面来看从工作线程得到结果时会发生什么。应该记得，创建工作线程时，我们指定了一个消息处理程序调用**processWork**：

```
var worker = new Worker("worker.js");

worker.onmessage = function(event) {
   processWork(event.target, event.data);
}
```

这个消息处理程序要调用processWork，将来自工作线程的数据传递给这个函数，同时还要提供目标，目标就是发出数据的工作线程的引用。

工作线程向我们发回消息来提供结果时，要由这个processWork函数处理结果。可以看到，向这个函数传入了两样东西：消息的目标和消息的数据（也就是任务对象再加上对应图像中一行的值）。所以我们现在的任务就是编写processWork（把这个代码键入到mandel.js中）：

```
function processWork(worker, workerResults) {
    drawRow(workerResults);
    reassignWorker(worker);
}
```

将结果传递到drawRow，在画面上绘制这些像素。

现在这个工作线程的工作就完成了，所以可以为它重新分配另外一个任务。为此，要编写一个函数reassignWorker。

就快达到目标了，既然提到了reassignWorker，现在就来编写这个函数。它的工作是这样的：使用nextRow全局变量检查我们正在计算的行，只要还有要计算的行（可以通过查看画布中有多少行来确定），可以为工作线程分配一个新任务。否则，如果没有更多工作要做，可以简单地将工作线程的idle属性设置为true。把这个代码也输入到mandel.js中：

```
function reassignWorker(worker) {
    var row = nextRow++;

    if (row >= canvas.height) {
        worker.idle = true;
    } else {
        var task = createTask(row);
        worker.idle = false;
        worker.postMessage(task);
    }
}
```

为这个工作线程指定需要计算的下一行，所以从nextRow得到行号，并让nextRow递增（这样下一个工作线程就会得到下一行）。

如果这一行大于或等于画布的高度，说明已经大功告成！我们已经用Mandelbrot集工作线程计算的结果填充了整个画布。

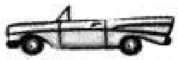
canvas是一个全局变量，在init函数中调用setupGraphics时设置。

不过，如果还有其他行要计算，就要为下一个工作线程创建一个新任务，确保这个工作线程的idle属性为false，并向它发送一个消息分配这个新任务。

这个代码放在mandel.js文件中。

pin5i.com 拼吾爱

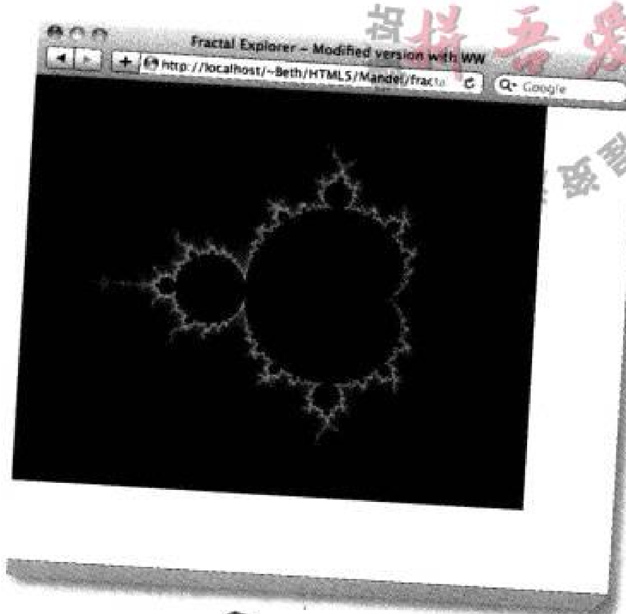
梦幻测试



代码已经够多了！下面来测试一下。在浏览器中加载fractal.html文件，看看你的工作线程如何工作。取决于你的机器，你的Fractal Explorer应该比以前稍快一些。

我们还没有编写代码来处理浏览器窗口的大小调整，也没有处理点击来放大分形图像。所以目前你能看到的就是右边的这个图像。

不过，嘿，到目前为止还不错，是吧？



看到了！不过不能放大，这可不好，另外还没有占满整个窗口，这也不太好，不过下面就来解决这些问题……

处理点击事件

我们已经让工作线程行动起来，开始计算Mandelbrot集，并向我们返回结果，从而能绘制在画布上，但是如果点击放大会发生什么呢？幸运的是，由于我们使用了工作线程在后台完成大强度的计算，所以用户界面 可以很利索地处理你的点击。不过，还需要编写一些代码来具体处理点 击事件。我们的做法如下：

- [x] 创建HTML
- [x] [illegible]代码
- [x] 创建工作线程
- [x] 启动工作线程
- [x] 实现工作线程
- [x] 处理结果
- [ ] **用户交互代码**

① 首先需要增加一个处理程序，负责处理鼠标点击，要记住，点击事件发生在画布元素上。为此，只需要为画布的onclick属性增加一个处理程序，如下：

```
canvas.onclick = function(event) {
    handleClick(event.clientX, event.clientY);
};
```

如果点击了画布，就会调用函数handleClick，并提供点击的x和y位置。

将这个代码增加到"mandel.js"文件init函数中的setUpGraphics调用下面。

② 现在只需要编写handleClick函数。在具体编写之前，先来考虑一下：用户点击画布时，这说明他们希望放大所点击的那部分区域（可以退回到前面的单线程版本
http://wickedlysmart.com/hfhtml5/chapter10/singlethread/fractal.html
看看具体是怎么做的）。所以，用户点击时，我们需要得到他们想要放大的位置坐标，然后让所有工作线程投入工作，开始创建一个新图像。另外还要记住，我们已经有一个函数为所有空闲工作线程分配新工作：startWorkers。下面就来试一试……

pin51.com 拼吾爱 最新最全的下载站

用户点击画布放大分形图像时，会调用handleClick。

传入点击的x, y位置，从而知道用户点击了屏幕的哪个位置。

```
function handleClick(x, y) {
    var width = r_max - r_min;
    var height = i_min - i_max;
    var click_r = r_min + width * x / canvas.width;
    var click_i = i_max + height * y / canvas.height;

    var zoom = 8;

    r_min = click_r - width/zoom;
    r_max = click_r + width/zoom;
    i_max = click_i - height/zoom;
    i_min = click_i + height/zoom;

    startWorkers();
}
```

这个代码会调整所计算的分形区域大小，将x, y位置作为新区域的中心。另外还会确保这个新区域与现有图像有相同的宽高比。

这里设置了为工作线程创建任务所用的全局变量：放大级别（zoom）确定分形图像的放大程度，这会确定要计算Mandelbrot集的哪些值。

startWorkers(); ← 现在准备重新启动工作线程。

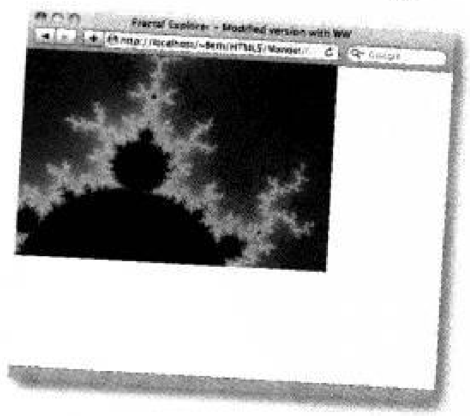
这个代码放在mandel.js文件中。

再做个测试

下面来测试以上所做的代码修改。在浏览器中重新加载fractal.html，这一次点击画布上的某个地方。点击后，你会看到工作线程开始创建一个放大的视图。

嘿，现在你应该可以开始探险了！尝试几次之后，下面再做最后几处修改，让这个实现尽善尽美。

不错！可以放大了，不过还需要调整画布大小，让它完全占满我们的窗口。



- [x] 创建HTML
- [x] [illegible]
- [x] 创建工作线程
- [x] 启动工作线程
- [x] 处理结果
- [x] 处理结果
- [ ] 用户交互代码

让画布占满浏览器窗口

我们说过，希望这个分形图像占满浏览器窗口，这说明如果窗口大小改变，也需要调整画布的大小。不仅如此，如果修改了画布大小，还应当向工作线程分配一组新的任务，让它们重新绘制分形图像，占满这个新的画布大小。下面就来编写代码，将画布的大小调整为浏览器窗口大小，完成之后还要重新启动工作线程。

```
function resizeToWindow() {
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;
    var width = ((i_max - i_min) * canvas.width / canvas.height);
    var r_mid = (r_max + r_min) / 2;
    r_min = r_mid - width/2;
    r_max = r_mid + width/2;
    rowData = ctx.createImageData(canvas.width, 1);

    startWorkers();
}
```

resizeToWindow确保将画布宽度和高度设置为与窗口的新宽度和高度一致。

还要根据新的宽度和高度更新工作线程用来完成计算的值（要确保分形图像总是占满画布，而且保持窗口的宽高比）。

还有一个管理方面的细节，我们使用了一个前面没有讨论过的全局变量：rowData。rowData是一个ImageData对象，要用来在画布上绘制一行像素。所以，调整画布大小时，需要重新创建rowData对象，使它的宽度与画布的新宽度相同。查看mandellib.js中的函数drawRow，了解如何使用rowData在画布上绘制像素。

再次重新启动工作线程。

现在需要再做一件事情：调用`resizeToWindow`作为浏览器窗口大小调整事件的处理程序。做法如下：

```
window.onresize = function() {
    resizeToWindow();
};
```

可以把这个代码放在`mandel.js`的init函数中，就放在`setUpGraphics`调用下面。

这个代码放在mandel.js文件中。

pin51.com 拼吾爱 最新最全的资源分享下载站

吹毛求疵的程序员

还有一个问题，虽然我们可以不去管它，但是如果没有它，代码看上去可能不太对劲。我们一起来考虑：假设你正在让一组工作线程辛辛苦苦地处理它们的行，突然之间，用户放弃了，又一次点击屏幕来放大图像。这可不太妙，因为工作线程一直在努力工作，处理它们分管的各行，现在用户却想放弃，要改变整个图像，这会让之前的所有工作都白费。更糟糕的是，工作线程根本不知道用户点击了屏幕，仍然打算返回它们计算的结果。还有更糟糕的，主页面中的代码很乐意接收和显示这一行！虽然不像到了世界末日那么糟，但是如果用户调整了窗口大小，我们也会遇到同样的问题。

致编辑：很抱歉这里有点罗嗦，不过，嘿，再有几页就可以交稿了……

现在你可能不会注意到这个问题，因为并没有太多的工作线程，而且这些工作线程可以很麻利地为新图像计算相同的行，覆盖原来那些不正确的行。不过，不管怎么说，就是感觉不太对劲。不仅如此，其实这个问题很容易修正，所以下面必须做出改进。

当然必须承认，我们原来就知道会有这个问题，你可能还记得之前加入了一个小变量generation。还记得吧，每次重启工作线程时都会让generation的值递增。另外应该还记得来自工作线程的results对象：每个结果都有“generation”属性。所以我们可以利用这个“代”，来了解是从当前这一次还是从前一次可视化得到的结果。

下面来看代码修正，然后再来讨论它是如何工作；编辑mandel.js中的processWork函数，增加下面这两行代码：

```
function processWork(worker, workerResults) {
    if (workerResults.generation == generation) {
        drawRow(workerResults);
    }
    reassignWorker(worker);
}
```

检查工作线程的结果，查看它的“代”与当前“代”是否一致。

如果确实一致，就绘制这一行，否则说明这个结果是旧的，将它忽略。

不论哪种情况，都会为工作线程重新分配新的工作！

所以，这里所做的就是先检查，确保我们处理的当前“代”与工作线程结果的“代”一致。如果是这样，那么太好了，只需要绘制这一行。如果不一致，这说明它肯定是旧的，所以只需将它忽略。工作线程为它浪费了时间固然很糟糕，不过我们可不希望在屏幕上绘制“老”图像中的数据行。

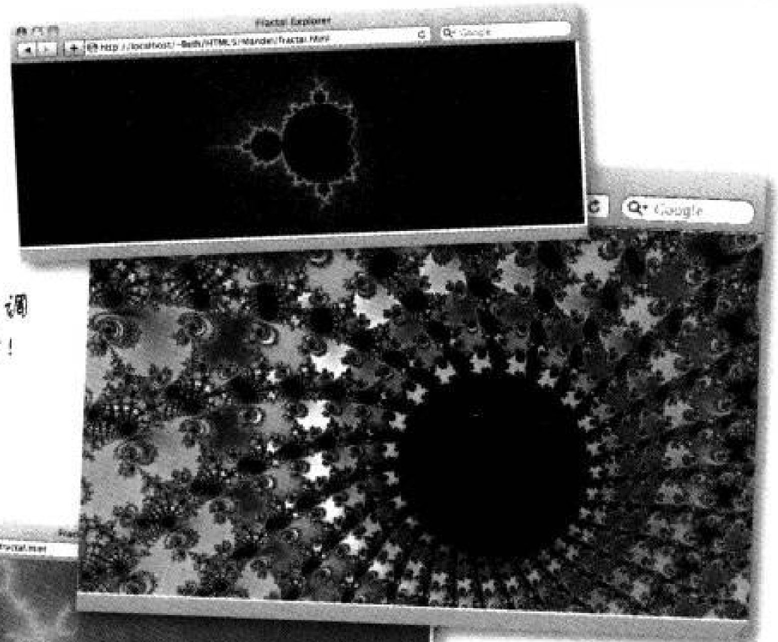
好了，终于完成了，千真万确，现在确保键入了以上修改，做好准备……

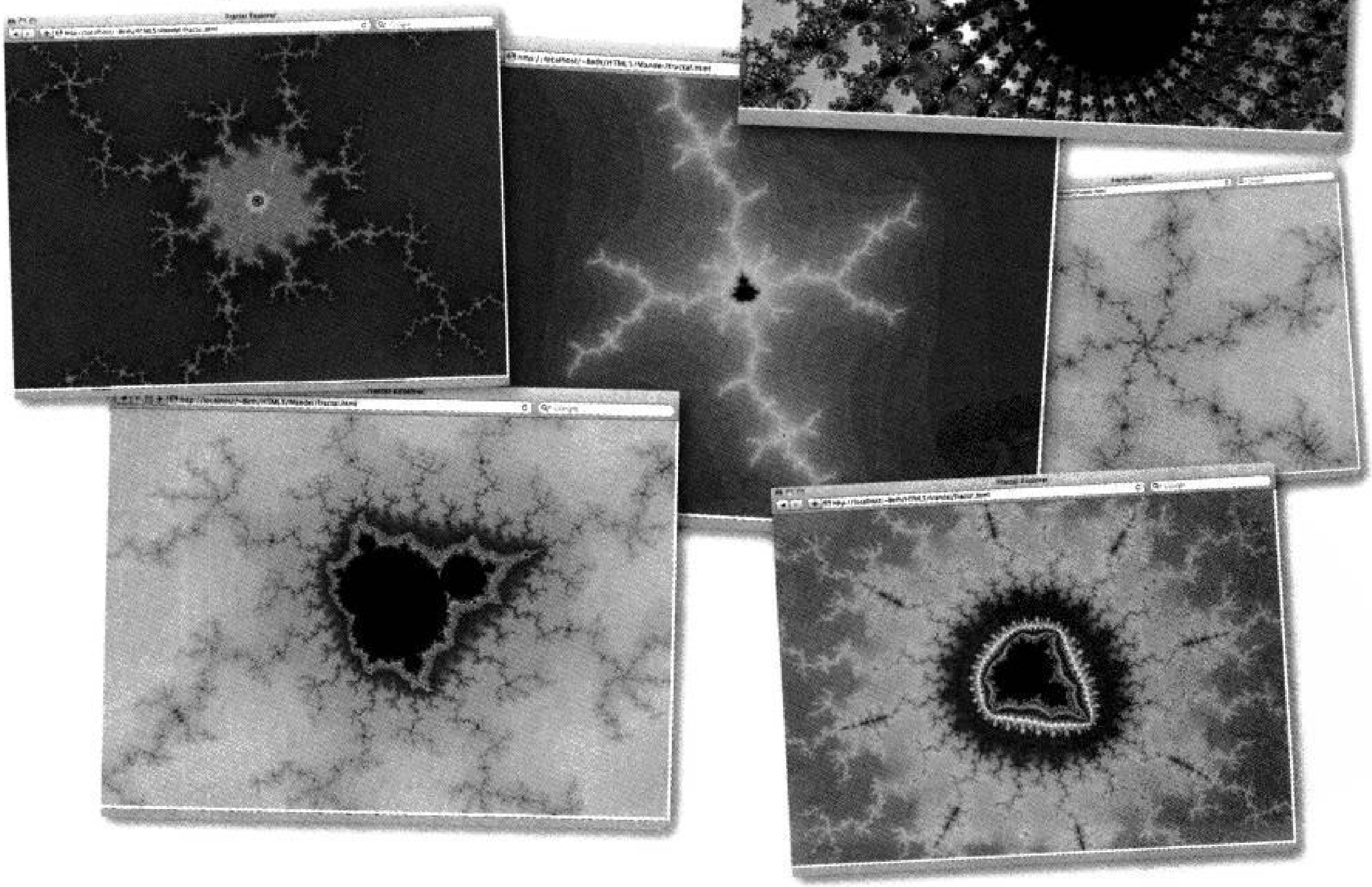
最后的测试！

大功告成了！现在你应该已经完成所有代码。将fractal.html文件加载到浏览器，看看你的工作线程如何开展工作。这个版本应该会比原来的单线程版本更快，而且更有响应性。如果你的计算机是多核的，速度会快得多。

现在可以放手尝试…… 放大…… 开始探险吧。如果你找到Mandelbrot集中未被发现的“国土”，请通知我们（如果你愿意，还可以通过微博把截屏图发给#hfhtml5）。

- [x] 创建HTML
- [x] [illegible]代码
- [x] 创建工作线程
- [x] 启动工作线程
- [x] 实现工作线程
- [x] 处理结果
- [x] 用户交互代码





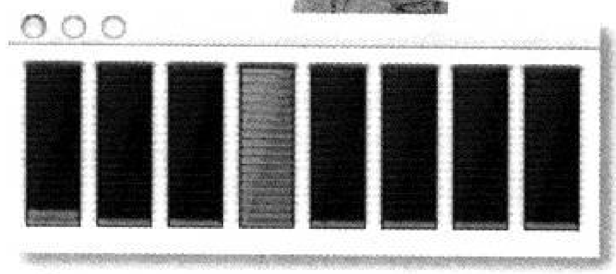
pin51.com 拼吾爱

实验室生活

如果你在编写高性能代码，可能想知道工作线程的个数对应用的运行时性能有什么影响。

为此，可以使用OS X或Windows上的任务监视器。如果退回到我们原来的版本（单线程版本，`http://wickedlysmart.com/hfhtml5/chapter10/singlethread/fractal.html`），性能可能如右图所示。



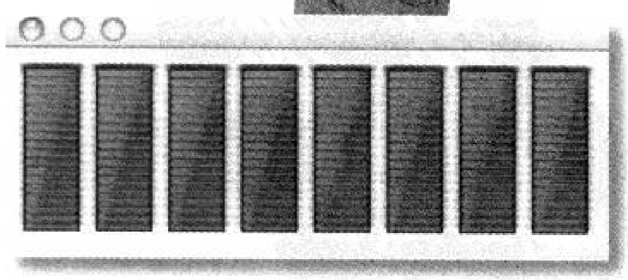


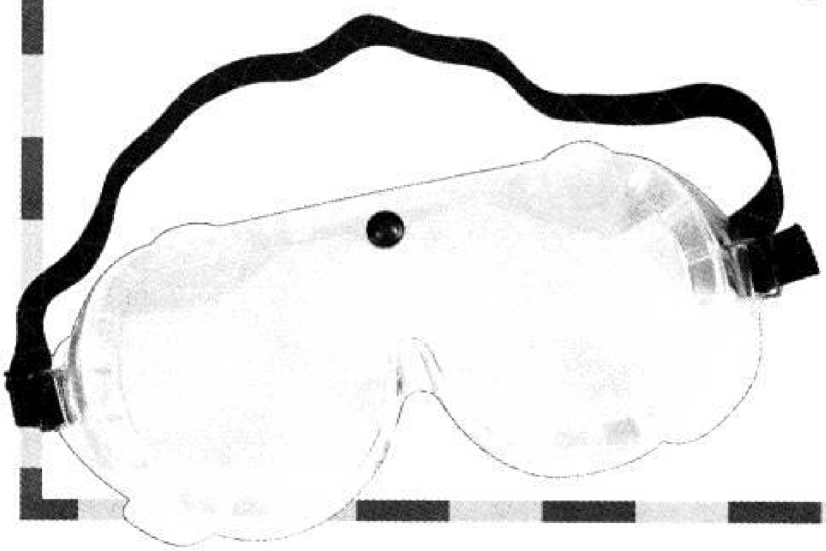
我们的机器上有8个核，为了与之匹配，所以在使用了Web工作线程的Fractal Explorer中，我们将工作线程个数也设置为8，即`numberOfWorkers = 8`。在活动监视器中可以看到，所有8个核都得到最大程度的利用。

如果把工作线程的个数设置为2，或4，或16，或者32，你认为会发生什么？或者设置为介于这些值之间的某个数呢？

可以在你的机器上试一试，看看哪些值对你来说最适合。

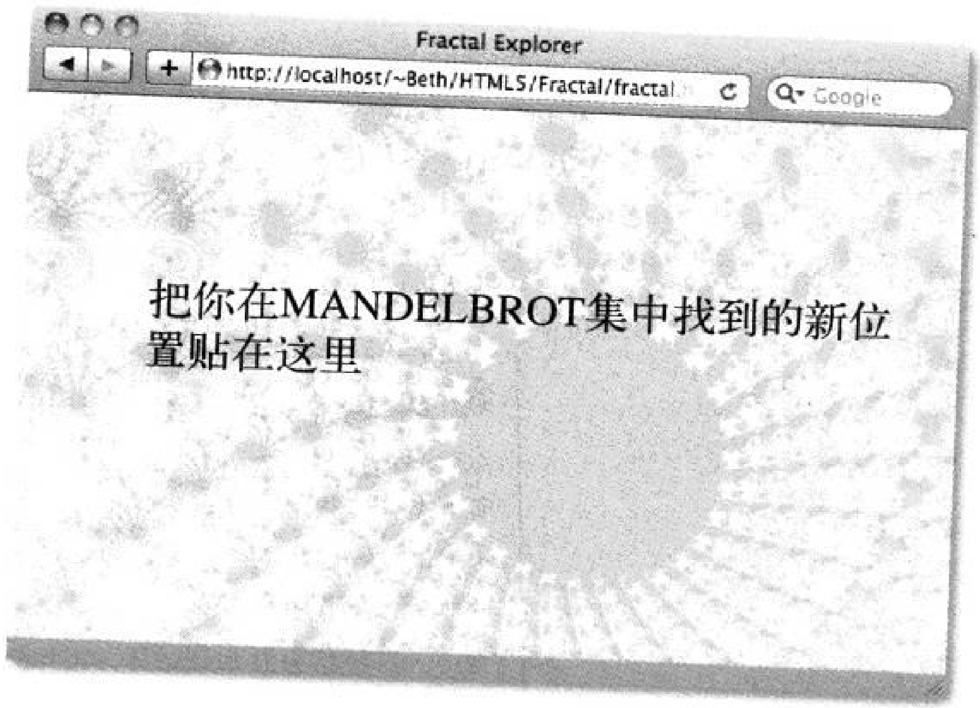




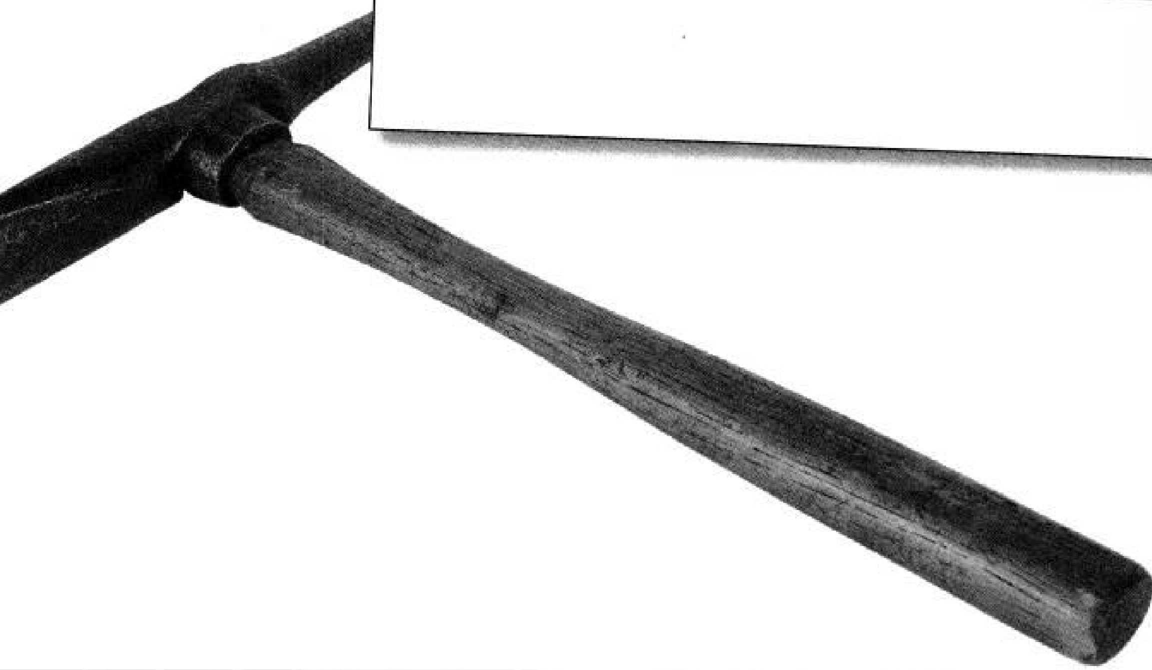


划定你的疆域!

终于完成了！你已经得到一个功能完备的Fractal Explorer，可以在Mandelbrot世界中放手探险了。还等什么呢，全力以赴找出属于你的那一小块虚拟王国吧。一旦找到，把它打印出来，贴在这里，然后给这块新领土取个名字。



为你的新领土命名：____________________



pin51.com 拼吾爱

在离开之前，相信吗？关于Web工作线程还有一些知识需要了解！请查看后面几页，了解这一章未涵盖的内容。



终止工作线程

你已经创建了工作线程来完成一个任务，任务完成时，你可能想清除所有工作线程（它们确实会占用浏览器中宝贵的内存）。可以在主页面中由代码终止工作线程，如下：

```
worker.terminate();
```

如果工作线程恰好正在运行，工作线程脚本会异常中止，所以使用时要当心。另外，一旦终止了一个工作线程，就无法再重用了。你必须创建一个新的工作线程。

还可以通过（从工作线程内部）调用`close();`让一个工作线程停止工作。

处理工作线程中的错误

如果工作线程中出现了严重的错误会怎么样？如何调试？可以使用onerror处理程序来捕获错误，还可以得到调试信息，如下：

```
worker.onerror = function(error) {
    document.getElementById("output").innerHTML =
        "There was an error in " + error.filename +
        " at line number " + error.lineno +
        ": " + error.message;
}
```

最新最全资源下载站 www.Java51.com 我爱书

使用importScripts建立一个JSONP请求

不能插入新的<script>元素从工作线程建立JSONP请求，不过可以使用importScripts来建立JSONP请求，如下所示：

```
function makeServerRequest() {
        importScripts("http://SomeServer.com?callback=handleRequest");
}
function handleRequest(response) {
        postMessage(response);
}
makeServerRequest();
```

还记得JSONP吧？在URL查询中包含回调函数，调用时就会把JSON结果传递到response参数。

工作线程中使用setInterval

你可能遗漏了这一点（我们的进度太快，只有一个例子中使用了这个特性），不过要知道，可以在工作线程中使用setInterval（和setTimeout）反复完成同样的任务。例如，可以更新"名言"工作线程(quote.js)，让它每3秒发送一个随机的名言，如下：

```
var quotes = ["I hope life isn't a joke, because I don't get it.",
              "There is a light at the end of every tunnel...just pray it's not a train!",
              "Do you believe in love at first sight or should I walk by again?"];
function postAQuote() {
    var index = Math.floor(Math.random() * quotes.length);
    postMessage(quotes[index]);
}
postAQuote();
setInterval(postAQuote, 3000);
```

把这两行移到一个postAQuote函数中……

……调用postAQuote立即发出一个名言，然后设置一个间隔来发送更多名言，每3秒发送一个。

子工作线程

如果工作线程完成任务时也需要帮助，可以创建它自己的工作线程。假设你为工作线程分配了要处理的图像区域，这个工作线程可以做出判断，倘若区域大于某个大小，它就会把这个任务分解到自己的子工作线程来完成。

工作线程创建子工作线程的做法就类似于页面代码中创建工作线程，如下：

```
var worker = new Worker("subworker.js");
```

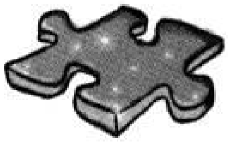
要记住，子工作线程与工作线程类似，也是重量级的。它们会占用内存，并作为单独的线程运行。所以，对于创建多少个子工作线程一定要当心。

pin5i.com 拼吾爱 最新最全下载站

BULLET POINTS

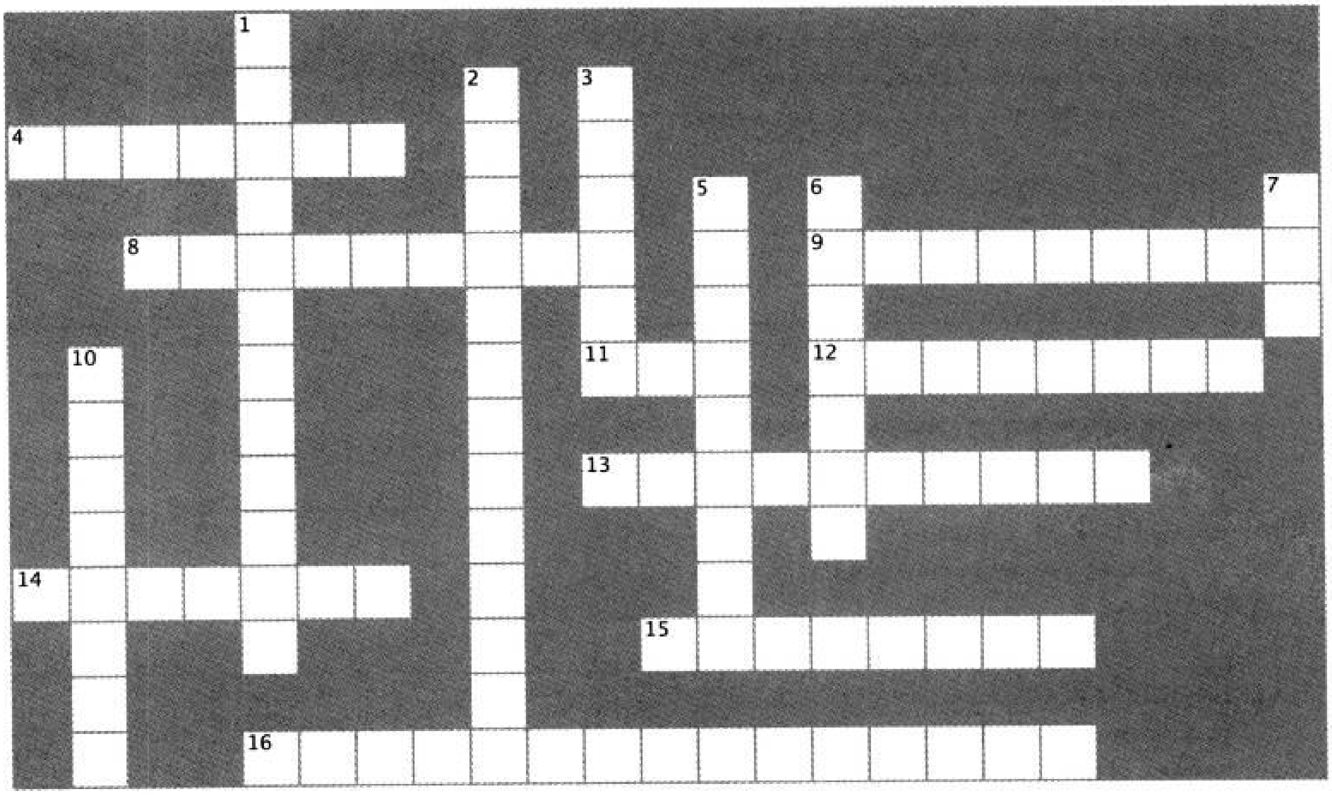
- 如果没有Web工作线程，JavaScript是单线程的，这说明它一次只能做一件事情。
- 如果交给一个JavaScript程序太多的工作，你可能会收到一个“slow script”（脚本运行缓慢）对话框。
- Web工作线程在一个单独的线程处理任务，所以主JavaScript代码可以继续运行，用户界面可以保持响应。
- Web工作线程的代码放在与页面代码不同的一个单独的文件中。
- Web工作线程不能访问页面代码中的函数或DOM。
- 页面中的代码与Web工作线程通过消息通信。
- 要向一个工作线程发送消息，可以使用postMessage。
- 可以通过postMessage向工作线程发送字符串和对象，但不能向工作线程发送函数。
- 可以将工作线程的onmessage属性设置为一个处理函数，来接收由工作线程返回的消息。
- 工作线程将其onmessage属性设置为一个处理函数，来接收页面代码发送的消息。
- 一个工作线程准备发回一个结果时，会调用postMessage，并传入结果作为参数。
- 工作线程结果封装在一个事件对象中，并置于data属性中。
- 可以使用event.target属性查找哪个工作线程发出了消息。
- 消息在主页面代码和工作线程之间会复制，而非共享。
- 可以使用多个工作线程完成能分解为多个任务的大规模计算，如计算一个分形可视化图像或对光线跟踪图像。
- 每个工作线程在它自己的线程中运行，所以如果你的计算机有一个多核处理器，工作线程会并行运行，这会提高计算的速度。
- 可以从页面代码调用worker.terminate()来终止一个线程。这会中止工作线程脚本。工作线程还可以调用close()让自己停止工作。
- 工作线程还有一个onerror属性。可以把这个属性设置为一个错误处理函数，如果你的工作线程存在一个脚本错误就会调用这个处理函数。
- 要在工作线程文件中包含和使用JavaScript库，可以使用importScripts。
- 还可以使用importScripts来利用JSONP。要在工作线程文件中实现传入URL查询的回调。
- 工作线程不能访问DOM或主代码中的函数，但是可以使用XMLHttpRequest和本地存储。

pin51.com 拼吾爱 最新最全的资源分享下载站



HTML5填字游戏

哇，已经到第10章了，终于完成了。坐下来，放松一下，让你的左脑活动活动，把这些知识牢牢记住。下面是第10章的填字游戏。



横向

4. 可以使用postMessage向工作线程传递________。
8. 处理器同时完成多个工作的能力。
9. 用来注册一个处理程序来接收消息的属性。
11. 工作线程不能访问________。
12. 第一个例子使用了这个游戏。
13. 最著名的分形。
14. _______/工作线程。
15. Mandelbrot周边一个美丽的区域称为__________谷。
16. 编写第一版分形浏览器的人。

对了，我们还没有告诉你，他的名字叫James Henstridge。

纵向

1. 工作线程可以使用XMLHttpRequest，还可以访问____________。
2. 如何将额外的代码导入一个工作线程。
3. 执行______。
5. 如何中止一个工作线程。
6. Mandelbrot使用___________数。
7. 如何创建工作线程。
10. 管理器和工作线程通过这些通信。

pin51.com 拼吾爱



扮演浏览器答案

现在假装你是执行JavaScript的浏览器。

```
window.onload = function() {
    var worker = new Worker("worker.js");
    worker.onmessage = function(event) {
        alert("Worker says " + event.data);
    }
    for (var i = 0; i < 5; i++) {
        worker.postMessage("ping");
    }
}
```

这会向工作线程发出5个ping消息，它会响应5个pongs，所以，我们会得到5个"Worker says pong"提醒。

```
window.onload = function() {
    var worker = new Worker("worker.js");
    worker.ommessage = function(event) {
        alert("Worker says " + event.data);
    }
    for(var i = 5; i > 0; i--) {
        worker.postMessage("pong");
    }
}
```

这会向工作线程发出5个pong消息，工作线程将忽略这些消息，因为它们不是ping。没有输出。

```
window.onload = function() {
    var worker = new Worker("worker.js");
    worker.onmessage = function(event) {
        alert("Worker says " + event.data);
        worker.postMessage("ping");
    }
    worker.postMessage("ping");
}
```

这会发出一个ping，然后每次返回一个pong时，再发送另一个ping，所以我们会进入一个无限循环，不停地出现pong提醒。

```
window.onload = function() {
    var worker = new Worker("worker.js");
    worker.onmessage = function(event) {
        alert("Worker says " + event.data);
    }

    setInterval(pinger, 1000);

    function pinger() {
        worker.postMessage("ping");
    }
}
```

这会每秒发出一个ping，所以每次发出一个ping时我们就会得到一个pong。

下载站 最新 拼吾爱 资源 www.pinwu51.com

Sharpen your pencil Solution

虽然工作线程一般通过消息得到工作号令，但并非必须如此。下面给出一种简洁的方法，可以让工作线程和HTML漂亮地完成工作。如果你看懂了这个代码要做什么，请在下面给出描述。

quote.html

```
<!doctype html>
<html lang="en">
  <head>
    <title>Quote</title>
    <meta charset="utf-8">
  </head>
<body>
  <p id="quote"></p>
  <script>
    var worker = new Worker("quote.js");
    worker.onmessage = function(event) {
      document.getElementById("quote").innerHTML = event.data;
    }
  </script>
</body>
</html>
```

quote.js

```
var quotes = ["I hope life isn't a joke, because I don't get it.",
          "There is a light at the end of every tunnel....just pray it's not a train!",
          "Do you believe in love at first sight or should I walk by again?"];
var index = Math.floor(Math.random() * quotes.length);

postMessage(quotes[index]);
```

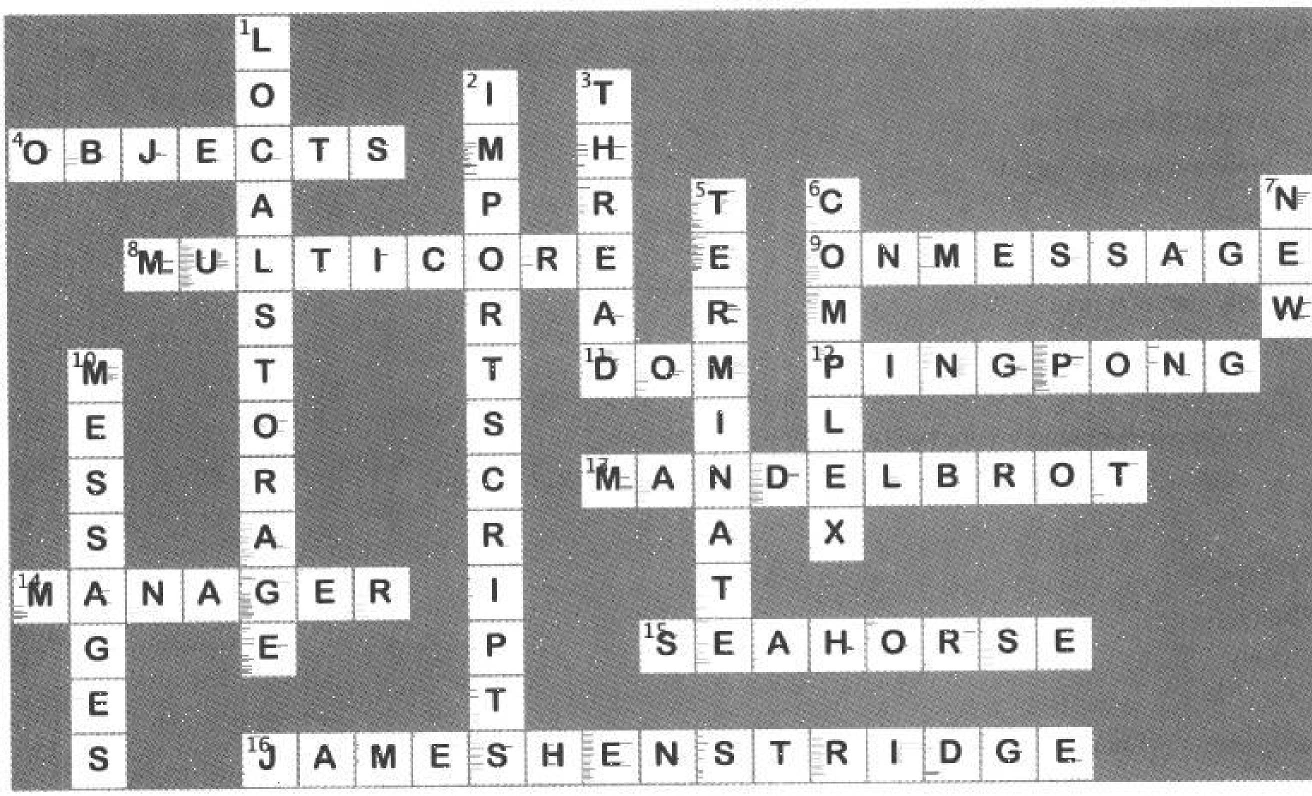
这里是你的描述：

在我们的HTML中，有一个脚本来创建工作线程，它会立即执行。工作线程从quotes数组随机地选择一个名言，并使用postMessage把这个名言发送给主代码。

主代码由event.data得到这个名言，把它增加到页面的“quote” <p>元素中。

pin51.com 拼吾爱 最新最全资源分享下载站

HTML5填字游戏答案



pin5i.com 拼吾爱



祝贺你！

终于到终点了。

当然了，后面还有一个附录。

还有索引。

还有封底。

然后还有网站……

实际上，这是无止境的。

pin5i.com 拼吾爱 最新编程资源分享下载站

附录：其他

(我们没有谈到的)十大主题

pin51.com 拼吾爱 最新最全资源分享下载站

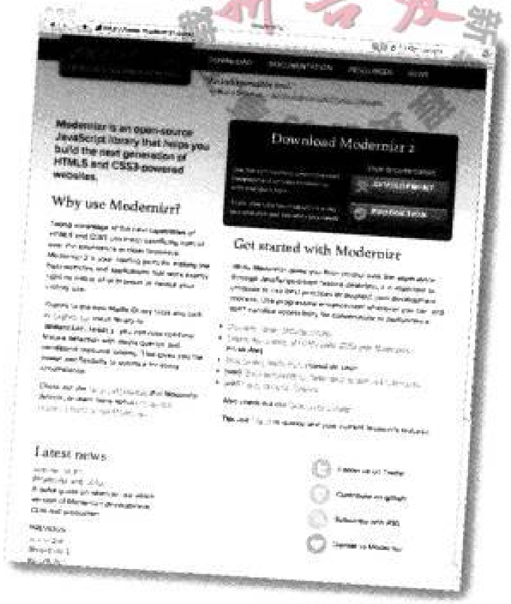


我们已经介绍了很多，这本书也快要结束了。我们会想你的，不过在你离开之前，还要再做一点准备，否则我们实在不放心让你贸然进入这个纷乱的世界。我们不可能把你需要知道的一切都放在这样短短的一章里。实际上，我们原先确实加入了（其他章节中没有谈到的）需要了解的有关HTML5的所有内容，只不过把字体缩小到0.00004。这样才能放得下，可惜没有人能看得清。所以，我们最后还是删去了大部分内容，只把最重要的部分保留在这个“十大主题”附录中。

这一回真的是本书最后一部分了，当然，除了索引以外（这也是必读的一部分）。

pin51.com 拼吾爱

#1 Modernizr



在这本书中，你可能已经注意到了，要想检测浏览器对一个API的支持，并没有统一的方法。实际上，几乎每一个API都会采用一种不同的方法检测。例如，对于地理定位API，我们会查找是否有一个geolocation对象作为navigator对象的属性，对于Web存储API，则要查看window对象中是否定义了localStorage，另外对于视频API，我们要查看是否可以在DOM中创建一个video元素，诸如此类。肯定还有更好的办法吧？

Modernizr是一个开源JavaScript库，提供了一个统一的接口来检测浏览器支持。Modernizer会负责不同检测方式的所有细节问题，甚至会考虑较老浏览器存在的所有边缘情况。Modernizr主页地址为`http://www.modernizr.com/`

Modernizr得到了众多开发人员的支持，所以你会看到Web上已经在广泛使用。我们强烈推荐使用这个库。

在页面中包含Modernizr

要使用Modernizr，需要在页面中加载这个JavaScript库。为此，首先要访问Modernizer网站（`http://www.modernizr.com/download/`），可以在这里定制配置一个库，其中只包含你需要的检测代码（或者，既然能得到所有检测代码，当然可以全部下载）。下载之后，把这个库存放在你选择的一个文件中，并把它加载到你的页面中（可以访问Modernizr网站，其中还提供了更多教程以及有关最佳实践的文档）。

如何检测支持

一旦安装了Modernizr，检测HTML5元素和JavaScript API就容易多了，也更为直接：

这个例子可以检测地理定位、Web存储和视频，所有检测都采用一致的方式。

说明：Modernizr并不只是用来完成API检测，还可以检测对CSS特性、视频编解码器和很多其他方面的支持。你可以试试看！

```
if (Modernizr.geolocation) {
    console.log("You have geo!");
}
if (Modernizr.localstorage) {
    console.log("You have web storage!");
}
if (Modernizr.video) {
    console.log("You have video!");
}
```

pin51.com 拼吾爱 最新最全电子书下载站

#2 音频

HTML5利用<audio>元素提供了一种在页面中播放音频的标准方法，而无需使用插件：

```
<audio src="song.mp3" id="boombox" controls>
  Sorry but audio is not supported in your browser.
</audio>
```

看着很熟悉？没错，音频支持的功能与视频类似（当然要逊于视频）。

除了<audio>元素，还有一个相应的音频API（Audio API），其中支持你希望有的一些方法，如play、pause和load。如果你觉得听上去很熟悉，没错，确实如此，因为音频API（在适当的方面）与视频API完全对应。音频还支持视频API中可以看到的很多属性，如src、currentTime和volume。下面给出一些音频代码，希望你能由此了解如何在页面中结合一个元素使用这个API：

```
var audioElement =
    document.getElementById("boombox");

audioElement.volume = .5;

audioElement.play();
```

得到音频元素的一个引用，然后将音量减半，开始播放。

同样类似于视频，每个浏览器实现的音频播放器控件（通常包括一个进度条，以及播放、暂停和音量控制控件）都有各自不同的外观。

除了简单的播放功能外，音频元素和API还提供了很多其他控制。与我们对视频的处理一样，可以隐藏音频控件，通过代码管理音频的播放来创建有趣的Web体验。利用HTML5，现在完全可以做到这一点，而不再需要使用（和学习）插件并因此引入开销。

音频编码的标准

很遗憾，同样与视频类似，音频也没有标准编码。目前有3种格式很流行：mp3、wav和Ogg Vorbis。你会发现，对这些格式的支持在不同浏览器上会有所不同，各种浏览器中对不同格式有不同程度的支持（写这本书时，Chrome是唯一一个同时支持这3种格式的浏览器）。



www.pin51.com 拼吾爱 最新最全电子书

#3 jQuery

jQuery是一个JavaScript库，它的目的是减少和简化处理DOM、使用Ajax，以及向页面增加视觉效果所需的JavaScript代码和语法。jQuery是一个相当流行的库，得到了广泛使用，而且可以通过其插件模型扩展。

要记住，Ajax就是使用XMLHttpRequest的另一个名字（类似第6章中的做法）。

现如今，用jQuery能做到的JavaScript也都能做到（我们说过，jQuery就是一个JavaScript库），不过jQuery能减少需要编写的代码，它在这一方面能力很强。

从jQuery的流行程度就可见一斑，不过如果你以前没有接触过jQuery，可能需要花些时间来习惯它的用法。下面来了解利用jQuery能做哪些事情，如果认为对你来说适用，建议再仔细地研究。

目前，jQuery工作经验是一项很好的技能，对于求职很有帮助，另外也有助于了解别人编写的代码。

对于初学者，还记得我们在这本书中写的所有window onload函数吗？如下：

```
window.onload = function() {
    alert("the page is loaded!");
}
```

使用jQuery也是一样：

```
$(document).ready(function() {
    alert("the page is loaded!");
});
```

与前面一样，文档准备好时就调用这个函数。

或者还可以进一步简化为：

```
$(function() {
    alert("the page is loaded!");
});
```

这很酷，不过可以看到，开始时可能还不习惯，需要一段时间才能习惯这种用法。不用担心，很快你就会发现这很自然。

那么从DOM得到元素呢？这正是jQuery的闪光之处。假设页面中有一个id为“buynow”的锚，希望为这个元素的点击事件指定一个点击处理程序（这本书中已经做过很多次了）。下面来看如何做到：

这里怎么回事？首先建立一个函数，它会在页面加载时得到调用。

```
$(function() {
    $("#buynow").click(function() {
        alert("I want to buy now!");
    });
});
```

接下来获取有一个id为“buynow”的锚（注意jQuery使用CSS语法选择元素）。

然后调用一个jQuery方法，对结果调用click来设置onclick处理程序。

pin51.com 拼吾爱 最新最全下载站

这只是一个开始，还可以很容易地对页面上的每一个锚设置点击处理程序：

```
$(function() {
    $("a").click(function() {
        alert("I want to buy now!");
    });
});
```

要做到这一点，只需要使用标记名。

如果使用JavaScript而不是jQuery，代码就会麻烦得多，你可以比较看看。

或者，还可以做更复杂的事情：

比如查找一个id为playlist的元素中的所有<li>子元素。

```
$(function() {
    $("#playlist > li").addClass("favorite");
});
```

然后把它们增加到类“favorite”。

实际上这只是一个jQuery热身，jQuery还可以做远比这里复杂得多的工作。

jQuery还有另外一个方面，可以对元素完成很多有趣的界面转换，如下：

```
$(function() {
    $("#specialoffer").toggle(function() {
        $(this).animate({ backgroundColor: "yellow" }, 800);
    },function() {
        $(this).animate({ backgroundColor: "white" }, 300);
    });
});
```

这会让id为specialoffer的元素在两个状态之间切换，一个是黄色、800像素宽，另一个是白色、300像素宽，并动画展现这两个状态之间的过渡。

可以看到，利用jQuery可以做很多事情，我们甚至还没有谈到如何使用jQuery与Web服务通信，还有处理jQuery的所有插件。如果你感兴趣，最好访问`http://jquery.com/`，参考其中提供的教程和文档。

还可以看看《Head First jQuery》！

www.pin51.com 拼吾爱 最新最全资源分享下载站

#4 XHTML死了，还是XHTML永存

这本书对XHTML的讨论很粗略，开始时先做了一个"XHTML死了"的讨论，然后简单讨论了"JSON与XML"。事实上，谈到XHTML，只是XHTML 2及以后版本没有生命力，实际上，如果你愿意，完全可以采用XHTML方式编写HTML5。为什么还希望这么做呢？嗯，你可能需要验证文档为XML，或者要把文档转换为XML，也可能希望结合HTML支持一些XML技术，如SVG（参见#5）。

下面来看一个简单的XHTML文档，然后再强调一些重点（我们不可能全面介绍关于这个主题需要知道的所有方面，与XML类似，这个内容很快就会变得相当复杂）。

```
<!DOCTYPE html>                                          ← 同样的doctype!
<html xmlns="http://www.w3.org/1999/xhtml">              ← 这是XML，需要一个命名空间！
   <head>
      <title>You Rock!</title>
      <meta charset="UTF-8" />     ← 所有元素必须是良构的，注意这里最后要有/>来结束这个空元素。
   </head>
   <body>
      <p>I'm kinda liking this XHTML!</p>
      <svg xmlns="http://www.w3.org/2000/svg">   ← 使用SVG在页面上绘制一个矩形。有关SVG的更多内容参考#5（下一页）。
         <rect stroke="black" fill="blue" x="45px" y="45px"
               width="200px" height="100px" stroke-width="2" />
      </svg>                       ↑ 可以把XML直接嵌在页面中！很酷哦！
   </body>
</html>
```

对于你的XHTML页面还有几点需要考虑：（所有元素都必须有结束标记，属性值两边要加引号，元素要正确地嵌套，诸如此类。）

- 页面必须是良构的XML。
- 页面要以`application/xhtml+xml` MIME类型提供，为此需要确保服务器提供这种类型的页面（可以研究有关的文档，或者咨询服务器管理员）
- 确保`<html>`元素中包含XHTML命名空间（上面就做到了这一点）。

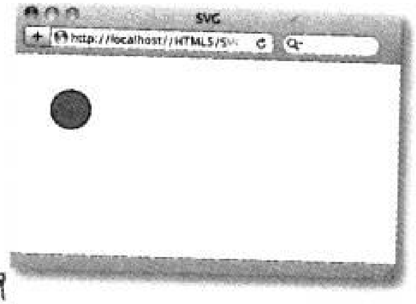
正如我们所说，关于XML有很多东西需要了解，另外有很多方面需要注意。一如往常，对于XML，愿主赐予你力量……

pin51.com 拼吾爱

#5 SVG

除了画布之外，还可以用另一种方法在web页面中自然地加入图形，这种方法就是可缩放矢量图形（Scalable Vector Graphics）或SVG。SVG已经存在一段时间了（大约1999年左右就有了），现在所有主流浏览器的当前版本都提供支持，也包括IE9及以后版本。

你知道的，画布是一个元素，允许用JavaScript在页面的一个位图绘制表面绘制像素，SVG则与画布不同，SVG图形要用XML指定。“XML?” 你可能很诧异，没错，就是XML! 你要创建表示图形的元素，然后可以采用复杂的方式把这些元素结合起来，构成图形场景。下面就来看一个非常简单的SVG例子：



这里使用XHTML形式的HTML5，因为我们要使用SVG，它基于XML。

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>SVG</title>
  <meta charset="utf-8" />
</head>
<body>
  <div id="svg">
    <svg xmlns="http://www.w3.org/2000/svg">
      <circle id="circle"
              cx="50" cy="50" r="20"
              stroke="#373737" stroke-width="2"
              fill="#7d7d7d" />
    </svg>
  </div>
</body>
</html>
```

在HTML中直接使用一个<svg>元素！

我们的SVG很简单。它只包含一个圆，位于x=50, y=50的位置，半径为20……

……笔划线宽为2像素宽，颜色为深灰……

……用中灰填充。

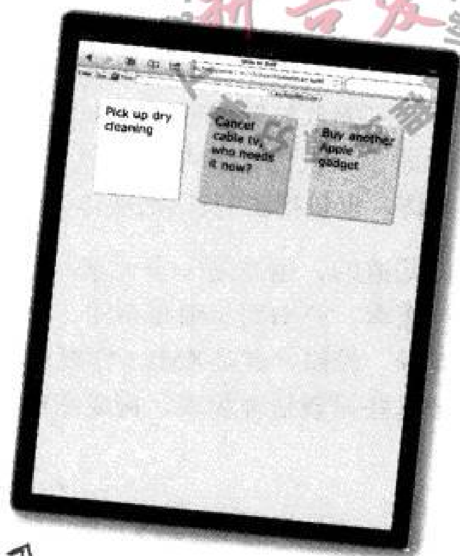
就像所有其他元素一样，可以从DOM获取这个圆元素，并对它进行处理……，例如，可以增加一个点击处理程序，当用户点击这个圆时，将圆的fill属性改为“red”。

SVG定义了大量基本形状，如圆、矩形、多边形、线条等。如果你要绘制更复杂的形状，还可以用SVG指定路径。当然，那时情况会变得更复杂（由画布中的路径就可以看出）。不过，有些图形编辑器允许你绘制一个场景，然后把它作为SVG导出，这样你就不用为自己确定所有这些路径而发愁了！

SVG有什么好处？嗯，SVG的一个好的方面是你可以随意地放大和缩小图形，它们不会像素化，而缩放jpeg或png图像时就存在像素化的现象。这样一来，SVG图像就能在不同情况下很容易地重用。另外，由于SVG用文本指定，所以可以搜索、索引和压缩SVG文件，还可以指定脚本。

我们只是粗略地介绍了使用SVG可以做什么，如果你对这个主题感兴趣，还可以进一步深入研究。

#6 离线Web应用



利用离线Web应用，没有连网的时候也可以使用你喜欢的Web应用！

如果你有一个智能手机或者平板电脑，可能会在外出时访问Web，利用WiFi和蜂窝网络，我们几乎无时无刻都能上网。不过，倘若确实不能上网该怎么办呢？你已经为自己构建了这些绝妙的HTML5 Web应用，如果一直都能使用这些应用那该多好！

嗯，对目前来说，这是可以做到的。所有现代桌面和移动浏览器都支持离线Web应用（只有一个例外：IE）。

那么如何让你的Web应用离线也能使用呢？可以创建一个缓存清单文件（cache manifest file），其中包含应用需要使用的所有文件的一个清单，浏览器会下载所有这些文件，如果你的设备离线使用，就会切换到使用这些本地文件。要告诉Web页面有这样一个清单文件，只需要把这个缓存清单文件的文件名增加到`<html>`标记，如下：

```
<html manifest="notetoself.manifest">
```

这里的`notetoself.manifest`文件包含以下内容：

```
CACHE MANIFEST
CACHE:
notetoself.html
notetoself.css
notetoself.js
```

每个缓存清单文件必须以这一行开头。

在CACHE节中列出你想要缓存的所有文件：HTML、CSS、JavaScript、图像等。

这里指出：访问指向这个文件的Web页面时，要下载CACHE节中所列的所有文件。还可以向这个文件增加另外两节：`FALLBACK`和`NETWORK`。FALLBACK指定试图访问一个文件但该文件未缓存时要转而使用什么文件，`NETWORK`指定不应缓存的文件（例如，访问跟踪资源）。

在着手尝试之前，需要知道两点：首先，需要确保Web服务器得到适当地设置，能够为缓存清单文件正确地提供相应的MIME类型（类似于第8章对视频文件的处理）。例如，在一个Apache服务器上，要在顶级Web目录的`.htaccess`文件中增加下面这行代码：

```
AddType text/cache-manifest .manifest
```

还有一点需要知道，测试离线Web应用相当困难！建议你查阅一本有关这个主题的好的参考书，另外还可以阅读HTML离线Web应用规范。

实现了基本缓存之后，可以使用JavaScript来得到缓存事件的通知，如缓存清单文件更新以及缓存状态等等。要得到事件通知，需要为`Window.applicationCache`对象增加事件处理程序，如下所示：

实现错误处理程序，从而在缓存出错时得到通知。

```
window.applicationCache.addEventListener("error", errorHandler, false);
```

#7 Web Socket

这本书中我们已经了解了两种通信方法：XMLHttpRequest和JSONP。在这两种方法中，都使用了一个基于HTTP的请求/响应模型。也就是说，要使用浏览器做出请求来得到初始的Web页面、CSS和JavaScript，每次需要什么时，就会使用XMLHttpRequest或JSONP做出另一个请求。甚至在并不会向我们提供任何新数据时也会发出请求，糖果机例子中有时就会出现这种情况。

Web Socket是一个新增的API，允许与一个Web服务的连接保持打开，这样只要有新数据，Web服务就可以把数据发送给你（而且你的代码会得到通知）。可以把它想成是你与Web服务之间的一个接通的电话线。

可以从高层角度来了解如何使用Web Socket：首先，为了创建一个Web Socket，要使用Web socket构造函数：

注意这个URL使用ws协议，而不是http协议。

```
var socket = new WebSocket("ws://yourdomain/yourservice");
```

要记住，必须由你或其他人写这个服务器代码，否则没有可以通信的服务器！

只要这个Socket通过open事件打开，你就会得到通知，要为它指定一个处理程序：

```
socket.onopen = function(){
    alert("Your socket is now open with the web service");
}
```

这里提供了一个处理程序，socket完全打开并准备通信时就会调用这个处理程序。

可以用postMessage方法向Web服务发送一个消息：

```
socket.postMessage("player moved right");
```

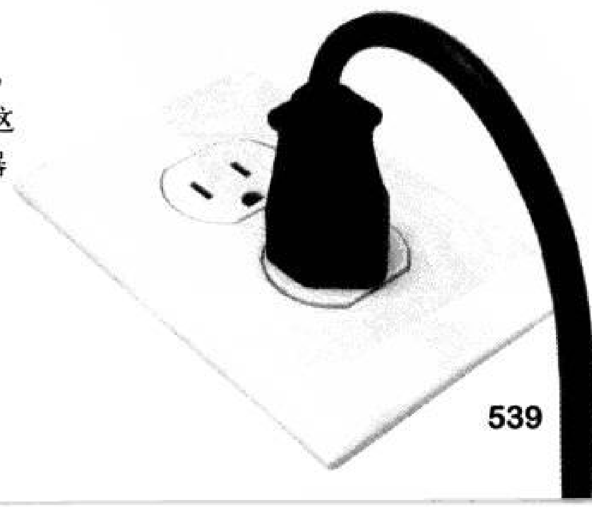
这里向服务器发送一个字符串，将来还可以发送二进制，不过二进制消息还没有得到广泛的支持。

为了接收消息，需要注册另一个处理程序，如下：

```
socket.onmessage = function(event) {
    alert("From socket: " + event.data);
};
```

通过注册一个处理程序，我们会接收所有消息，消息包含在事件的data属性中。

当然，除此以外，还有另外一些内容需要了解，可以参考一些在线教程，不过这个API并没有太多新内容。相对于其他一些HTML5 API的开发，这个API有些滞后，所以在接手一个重要项目之前，先要查看最新的浏览器兼容性指南。



www.pin51.com 拼吾爱 最新最全的资源分享下载站

#8 更多画布API

我们在第7章已经见识过画布，并用画布构建了我们的TweetShirt启动项目。不过除此以外，利用画布还可以做很多其他有趣的事情，我们希望在这里再简单介绍几点。

前面简要地提到过：可以保存和恢复画布上下文。为什么要这么做呢？假设你设置了上下文的一些属性，如fillStyle、strokeStyle、lineWidth等。然后你想临时改变这些值来完成某个工作，如绘制一个形状，但是做完这个工作之后不想再逐个地把它们重置为原先的属性值。这种情况下就可以使用save和restore方法来做到：

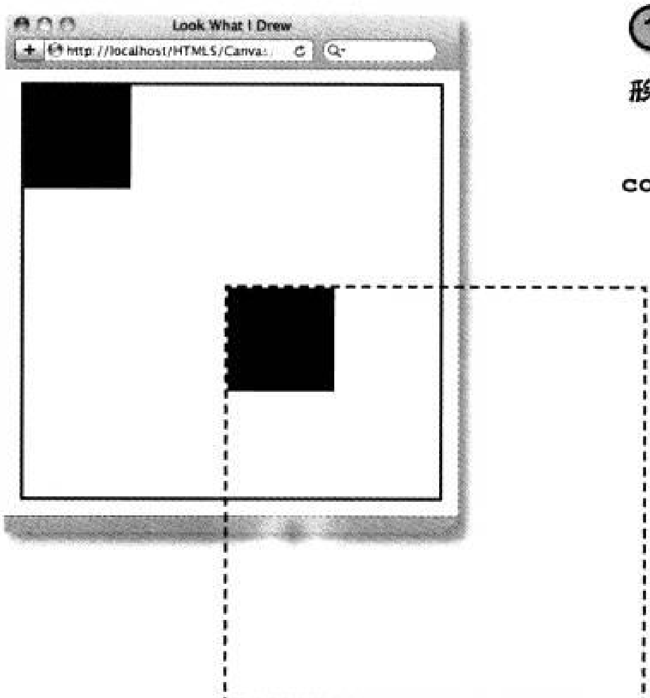
```
context.fillStyle = "lightblue";
 ...
context.save();
context.fillStyle = "rgba(50, 50, 50, .5)";
context.fillRect(0, 0, 100, 100);
context.restore();
 ...
```

我们在上下文中建立了一组属性，并完成一些绘制。

现在保存上下文。所有这些属性都会安全地保存下来。现在可以改变属性了……

……然后再把它们恢复到之前保存的属性值，只需调用restore方法！现在所有属性都会恢复为原先保存时的状态。

如果你想平移或旋转画布来完成一些绘制，然后再恢复到缺省位置，这种情况下这些方法尤其有用。translate和rotate方法有什么作用呢？下面就来看一看……



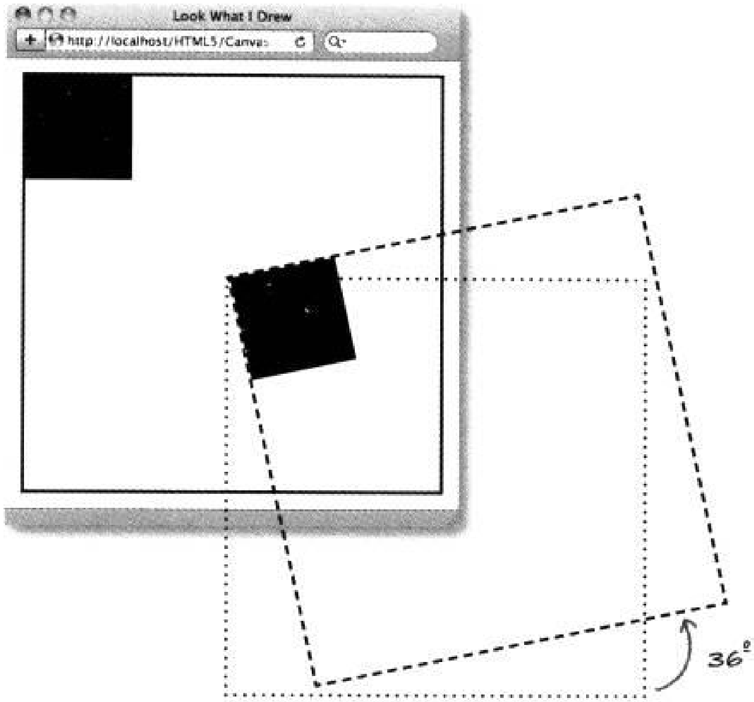
① 页面中有一个400×400的画布。如果在x=0, y=0的位置绘制一个黑色矩形，正如你期望的，它将绘制在左上角。

```
context.fillRect(0, 0, 100, 100);
```

② 现在，获取画布，将它向右、向下分别移动200像素。如果在x=0, y=0再画另一个矩形，这个矩形会绘制在前一个矩形向右、向下200像素的位置。这里就是对画布进行了平移。

```
context.translate(200, 200);
context.fillRect(0, 0, 100, 100);
```

pin51.com 拼吾爱



③ 如果在绘制矩形前旋转画布呢？（默认地）画布会绕着它的左上角旋转，由于我们刚才将左上角移动到200,200的位置，所以画布会绕这个位置旋转。

```
context.translate(200, 200);
context.rotate(degreesToRadians(36));
context.fillRect(0, 0, 100, 100);
```

平移或旋转画布时，它会在一个相对于浏览器窗口左上角定位的方格中移动。如果使用CSS确定画布的位置，需要考虑到这些值。可以试试看！

现在把所有这些内容综合起来！可以结合使用translate和rotate方法来创建一些有意思的效果。

```
var canvas = document.getElementById("canvas");
var context = canvas.getContext("2d");
var degrees = 36;

context.save();
context.translate(200, 200);
context.fillStyle = "rgba(50, 50, 50, .5)";

for (var i = 0; i < 360/degrees; i++) {
  context.fillRect(0, 0, 100, 100);
  context.rotate(degreesToRadians(degrees));
}
context.restore();
```

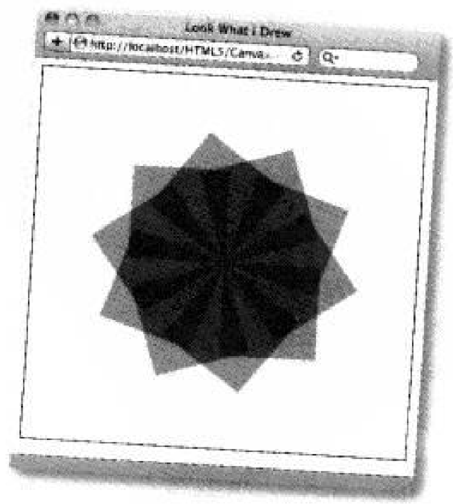
这里保存了上下文，以便在工作完成之后能很容易地恢复为正常的方格位置。

将画布平移200, 200。

每次循环时，在0,0绘制一个矩形之前将画布旋转36°，共绘制10个矩形。

现在我们的画布恢复为原来的位置！

这里是最后得到的结果。希望你喜欢！



这些简单的变换可以与其他更强大（也更复杂）的方法（如组合和变换）结合，用画布可以创建丰富多彩的平面艺术，只有想不到，没有做不到。

pin51.com 拼吾爱 最新最全电子书免费下载

#9 选择器API

你已经知道如何使用document.getElementById;从DOM选择元素，这本书中我们一直在用这种方法来结合HTML和JavaScript。另外你还知道如何使用document.getElementsByTagName（这个方法会返回与一个标记匹配的所有元素的数组），甚至还有一个getElementsByClassName方法（可以想象，这会返回一个给定类的所有元素）。

利用HTML5，则有了一种从DOM选择元素的新方法，这是受jQuery的启发。现在通过使用document.querySelector方法，可以在JavaScript中使用CSS的选择器（不过CSS中使用选择器指定样式）来选择DOM中的元素。

假设有以下这个简单的HTML：

```
<!doctype html>
<html lang="en">
<head>
  <title>Query selectors</title>
  <meta charset="utf-8">
</head>
<body>
  <div class="content">
    <p id="avatar" class="level5">Gorilla</p>
    <p id="color">Purple</p>
  </div>
</body>
</html>
```

仔细研究这个HTML的结构。我们将使用选择器API从页面选择元素。

有一个<div>元素，类为“content”，另外有两个<p>元素，分别有自己的id，其中一个元素的类为“level5”。

现在，使用选择器API请求“avatar” <p>元素：

```
document.querySelector("#avatar");
```

这与document.getElementId("avatar")基本上是一样的。下面使用元素的类来选择这个元素：

```
document.querySelector("p.level5");
```

现在使用标记名和类来选择这个元素。

还可以选择作为<div>子元素的<p>元素，如下：

```
document.querySelector("div>p");
```

或者甚至可以这样：

```
document.querySelector(".content>p");
```

在这里，使用一个子选择器来选择作为<div>子元素的<p>元素。默认地，它会选择第一个子元素。

如果实际上想得到<div>中的所有<p>元素，可以使用选择器API中的另一个方法querySelectorAll：

```
document.querySelectorAll("div>p");
```

现在我们得到了<div>的所有<p>子元素！

类似于getElementsByTagName，querySelectorAll会返回一个元素数组。仅此而已！这个API中只包含这两个方法。尽管这个选择器API规模很小，不过选择元素的功能很强大。

#10 不过，还有呢！



嗯，我们真希望把没有谈到的内容限制为十项，不过看起来还有很多都没有提及，在读后面的索引之前，我们再用一页介绍更多内容。如下所列（要记住，这些领域还在不断发展，不过相信你希望先有所了解，以备以后参考）：

索引数据库API和Web SQL

如果要在本地存储你的数据，而且想找一个比Web存储API更专业的方法，可以关注Web数据库领域。目前有两个方法最有竞争力：Web SQL和IndexedDB。颇有讽刺意味的是，在这二者之中，Web SQL得到了更广泛的支持，但是最近却被标准委员会废弃（这表示他们不再采纳它作为一个标准，你的下一个启动项目最好不要建立在Web SQL之上）。另一方面，IndexedDB还没有广泛实现，但是得到了Google和Firefox的支持。IndexedDB允许快速访问一个很大的索引数据集合，而Web SQL是一个小SQL引擎，在浏览器中运行。要时刻关注这些技术的动态，它们一直在快速变化！

拖放

Web开发人员一直在用jQuery完成拖放，现在这个功能已经内置于HTML5中。利用HTML5拖放API，可以指定要拖动的元素，把它投放在哪里，还可以指定JavaScript处理程序，从而可以得到拖放时各种事件的通知。要让元素可拖放，只需把`draggable`属性设置为true。几乎所有元素都可以拖动：包括图像、列表、段落等。可以监听类似`dragstart`和`dragend`等事件来定制拖动行为，甚至可以改变一个元素的样式，使它在拖动时有你希望的外观。利用dataTransfer属性可以随所拖动的元素同时发送一些数据，也可以通过event对象访问这个数据，来了解一些情况，比如元素是否在移动或复制。可以看到，利用HTML5的拖放，现在我们有了很多绝好的机会来建立新的用户界面交互。

跨文档消息传递

在第6章中，我们使用了一种名为JSONP的通信模式，来避开XMLHttpRequest存在的跨域通信问题。还有一种方法可以在文档之间通信，甚至是不同域中的文档。跨文档消息传递API指出，可以使用一个iframe元素向你加载的一个文档传送消息。这个文档甚至可以位于不同的域！目前还不会在你的iframe中加载任何文档，你只是希望确保它来自你信任的一个域，并设置它接收你的消息。不过从未来的发展来看，这将成为一种在两个HTML文档之间来回接收消息的方法。

还可以继续……

关于HTML5有一点很棒，还有很多很多新的功能正在以飞快的速度发展。这一页原本还有很多东西要讲，不过实在没有篇幅了。所以请时刻关注`http://wickedlysmart.com`，了解HTML5的所有最新动态！

pin51.com 拼吾爱 最新最全资源分享下载站



pin51.com 拼吾爱 最新最全资源分享下载站

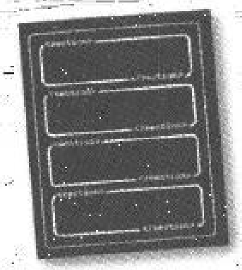
HTML5新构造指南

最近我们在Web镇为构建代码做了很多补充，而且为这些新构造准备了一个方便的指南，这可能是你一直想要的。具体来说，我们增加了一组新的语义元素，能够让你在构建页面时如虎添翼。目前我们的指南并不完备，实际上，这里的目标只是为你（有经验的构建人员）提供足够的信息，使你能熟悉这些新的HTML5元素和CSS属性，这样等你做好准备，学习如何构建本书中的Web应用时就可以使用这些新特性了。所以，如果你需要一个关于HTML5新增语义的快速教程，可以拿一份指南，这是完全免费的（不过时间有限，机不可失）。



Web镇HTML5语义元素指南

最近我们在Web镇对构建代码做了一些改变，还准备了一个便捷的指南供你参考。如果你一直在用<div>来表示常见的构造，如页眉、导航、页脚和博客文章，现在我们为你提供了一些新的构建模块。所以一定要熟悉这些新内容。



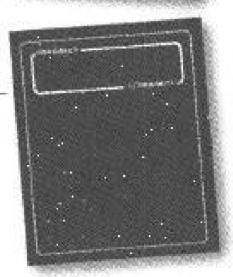
<section>

<section>是一个“通用文档”。可以使用<section>将文档（比如说，一个指南）标记为HTML。或者还可以包围HTML。<section>并不是一个通用容器，那是<div>的工作。另外要记住，只有对元素分组来指定样式时才使用<div>。



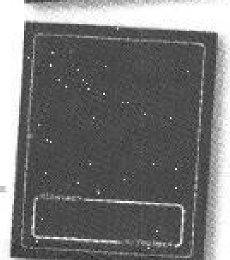
<article>

<article>是一个自包含的内容块，你可能希望与另一个页面或网站（甚至是你的狗狗）共享这个内容块。这个元素非常适用于博客文章和新闻。



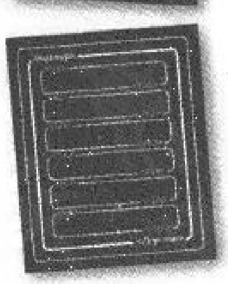
<header>

<header>放在<section>和<article>之类元素的最前面。还可以在体的最前面使用<header>为页面创建主页眉。



<footer>

<footer>放在其他元素的最下面。比如<section>、<article>和<div>。你可能以为一个页面上只能有一个<footer>，实际上不然，只要页面中某一节需要有脚注内容（如一篇文章的作者介绍或参考文献），都可以使用这个元素。

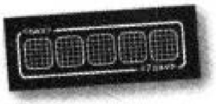


<hgroup>

这个元素可能有点难理解。不同于<header>，它可以包含与首部相关的任何元素，<hgroup>专用于归组<header>中的标题(<h1>...<h6>)。这对于建立大纲很有帮助。

pin51.com 拼吾爱

Web镇HTML5语义元素指南



<nav>

<nav>当然用于导航和链接。不过不只是单个链接，如果你有一组链接，也可以使用<nav>，比如网站的导航或者博客链接。不要用于段落中的链接。



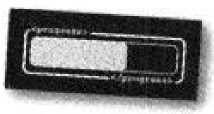
<aside>

<aside>很适合页面主结构之外的大块内容，比如旁注、引述或后来添加的内容。



<time>

终于到时间了！可以用<time>标记你的时间。慢慢来，要正确使用这个元素需要花些时间，你要好好研究一下<time>的合法格式。



<progress>

快完成了？没错，这些HTML5元素就快要介绍完了……<progress>表示完成一个任务的进度如何。使用这个元素，再加上一点CSS和JavaScript，可以得到一些很不错的效果。



<abbr>

嘿，拜托，如果单词很长就要使用缩写！这个元素很适合搜索，因为搜索引擎并不总是那么聪明，能够像我们一样知道这是缩写。



<mark>

可以使用<mark>标记单词，比如，用来突出显示或者完成编辑。这个元素很适用于搜索引擎结果。

pin51.com 拼吾爱

用CSS3为新构造增加样式

Web镇CSS3属性指南

既然已经准备好了构建模块，现在来考虑一些内部设计。你可能希望所有构造看上去都很美观，是不是？

新属性

CSS3中有很多新属性，其中很多属性所做的工作正是Web页面创作人员多年来一直苦心积虑通过HTML、图像和JavaScript才能办到的。例如：



```
opacity: 0.5;

border-radius: 6px;

box-shadow: 5px 5px 10px #373737;
```

让一个元素半透明。

在每个角上用一个6px的曲度创建一个圆角效果。

5px长、5px高，模糊度为10px，而且颜色为深灰的阴影。

新布局

有几种强大的新方法可以用CSS建立页面布局，这些方法远胜于定位，而且使用时要容易得多。例如：



```
display: table;
display: table-cell;
```

这会给出一个表格布局，而无需HTML表格。

```
display: flexbox;
flex-order: 1;
```

利用flexbox可以更好地控制浏览器如何在页面上摆放方框，如<div>。

新动画

利用动画，可以在属性值之间实现动画。例如，可以让某个元素消失，将透明度从不透明逐渐变为全透明：



```
transition: opacity 0.5s ease-in-out;

opacity: 0;
```

transition属性指定一个属性要渐变（这里指定了属性opacity），渐变所需的时间，以及松弛函数，所以变化是逐步完成的。

通过将opacity设置为0，（比如说在一个hover事件上），可以创建一个消失/重现动画。

新选择器

还有整个一组新选择器，包括nth-child，这允许指定一个元素中的特定子元素。最后，还可以设置一个列表中交替行的背景色，而且一点都不麻烦。

```
ul li:nth-child(2n) { color: gray; }
```

这表示要选择每一个间隔的列表项，并设置背景色为灰色。

pin5i.com 拼吾爱

索引

符号

$ (dollar sign)（$（美元符））
  $( ) (jQuery function)（$( )(jQuery函数)） 534
  beginning JavaScript variable names（作为JavaScript变量名的开头） 40, 42
2D drawing context（2D绘制上下文）, canvas（画布） 292. 参见 Canvas API; context, canvas
  defined（定义） 293
  getting（获取） 302
, (comma)（,（逗号））, separating object properties（分隔对象属性） 132
{ } (curly braces)（{}（大括号））
  enclosing code blocks（包围代码块） 26
  enclosing object properties（包围对象属性） 132
. (dot) operator（.(点)操作符）
  accessing object properties（访问对象属性） 133, 134
  invoking methods（调用方法） 151
// (forward slashes)（//（前向斜线））, beginning comments in JavaScript（开始JavaScript中的语句） 39
+ (plus sign)（+（加号））
  addition operator（加法操作符）or string concatenation operator（或字符串连接操作符） 45
  string concatenation operator（字符串连接操作符） 26
" " (quotation marks, double)（" "（引号，双引号））
  around JavaScript property values（JavaScript属性值两边） 133, 308
  around codecs parameter of <source> element（<source>元素codecs参数两边） 359
  denoting empty strings（指示空串） 26, 95, 108
  surrounding character strings in JavaScript（JavaScript中包围字符串） 39
; (semicolon)（;（分号））, ending statements in JavaScript（JavaScript中结束语句） 39
[ ] (square brackets)（[ ]（中括号））
  accessing and enumerating object properties（访问和列举对象属性） 133, 160
  associative arrays（关联数组） 424
  creating and indexing arrays（创建和索引数组） 67
  using with localStorage（用于localStorage） 424
_ (underscore)（_（下划线））, beginning JavaScript variable names（作为JavaScript变量名的开头） 40, 42

A

AAC Audio 357
<abbr> (abbreviation) element（<abbr>（缩写）元素） 547
accuracy（精度）, location information（位置信息） 191
  enableHighAccuracy option（enableHighAccuracy选项） 198
accuracy property（accuracy属性）, coordinates object（coordinates对象） 190
addEventListener method（addEventListener方法） 367
  calling error handler（调用错误处理程序） 406
  listener for ended video event（结束视频事件监听器） 386
  popping up play button after video ends（视频结束后显示播放按钮） 386
addition operator (+)（加法操作符 (+)） 45
addMarker function (example)（addMarker函数（示例）） 186
addStickyToDOM function (example)（addStickyToDOM函数（示例）） 430, 432, 440
  passing key as well as value each time it's called（每次调用时传递值以及值） 450
  using sticky object instead of string（使用sticky对象而不是字符串） 455
Adobe Premiere elements（Adobe Premiere元素） 360
Adobe's HTTP Dynamic Streaming（Adobe的HTTP Dynamic Streaming） 404
altitude and altitudeAccuracy properties（altitude和altitudeAccuracy属性）, coordinates object（coordinates对象） 190, 197
angles（角）
  measured in degrees（用度度量）, converting to radians（转换为弧度） 317
  startAngle and endAngle parameters of arc method（arc方法的startAngle和endAngle参数） 315
animations（动画）, new（新增）, in CSS3（CSS3中） 548
anonymous functions（匿名函数） 128
  using（使用） 129
Apache
  telling to serve video files with certain file extensions（通知提供有某些文件扩展名的文件） 371

using on Mac, PC, and Linux（Mac、PC和Linux上使用） 231
APIs (Application Programming Interfaces)（API（应用编程接口）） 15, 31
appendChild method（appendChild方法）
element object（element对象） 158
in addStickyToDOM function (example)（addStickyToDOM函数（示例）中） 450
ul object method（ul对象方法） 101
Apple's HTTP Live Streaming（Apple的HTTP Live Streaming） 404
application/xhtml+xml MIME type（application/xhtml+xmlMIME类型） 536
arc method（arc方法）, canvas context（画布上下文） 313
direction, startAngle, and endAngle parameters（direction, startAngle和endAngle参数） 315
drawing circles for t-shirt design app（为T恤设计应用画圆） 319
interpreting call to（解释调用）, and sketching out all parameters on circle（在圆上画出所有参数） 317, 343
using to trace a given path（用来跟踪给定路径） 316
x, y, and radius parameters（x, y和radius参数） 314
arguments（实参）, function （函数） 120
objects as（对象作为） 136
passing to parameters（传人行参） 122, 162
arrays（数组） 67
adding items（增加数组项） 68
creating and assigning to a variable（创建变量和赋值） 67
filling list items from (example)（从中填充列表项（示例）） 69
getting value of items in（获取数组项的值） 68, 303
length of（长度） 68
localStorage object as associative array（localStorage对象作为关联数组） 424
of objects（对象） 134, 457
removing items from（删除数组项） 73, 448
solving problems in local storage（解决本地存储中的问题） 439
storing in local storage（存储在本地存储中） 440, 467
using to store multiple values（用来存储多个值） 75
video playlist（视频播放列表） 365
of workers（工作线程） 508
<article> element（<article>元素） 546
<aside> element（<aside>元素） 547
associative arrays（关联数组） 424
attributes（属性）, gettin and settingg（获取和设置） 158
setting attribute on element（设置元素属性） 379
setting id attribute of sticky note（设置即时贴的id属性） 450
audio（音频） 16, 533
AAC and Vorbis encodings（AAC和Vorbis编码） 357
codecs 358
encodings in video files（视频文件中的编码） 356
formats（格式） 357, 533
methods and properties of audio API（音频API的方法和属性） 533
<audio> element（<audio>元素） 533
autoplay attribute（autoplay属性）, <video> element（<video>元素） 353, 354

B

background color（背景色）
canvas（画布）, filling before drawing new squares（绘制新方块前填充） 306, 342
setting backgroundColor property for sticky note（设置即时贴的backgroundColor属性） 455
setting for alternating rows in a list（为列表中的交替行设置） 548
background tasks（后台任务） 95
beginPath method（beginPath方法）, canvas context（画布上下文） 311, 319
"bitmap" drawing（"位图"绘制）, on canvas（画布上） 336
black and white（黑白）, converting pixels to（像素转换为） 399
<body> elements（<body>元素）, adding <script> elements（增加<script> 元素） to 53
booleans（布尔） 40
boolean expressions（布尔表达式） 43
conditional tests in for and while statements（for和while语句中的条件测试） 47, 49
using to make decisions with JavaScript（用来用JavaScript做决策） 49
true and false values（true和false值） 39
border-radius property（border-radius属性） 548
box-shadow property（box-shadow属性） 548

 element（
元素） 26
browsers（浏览器）
audio encodings support（音频编码支持） 533
background tasks（后台任务） 95
caching and repeated JSONP requests（缓存和重复JSONP请求） 272, 277

controls for HTML video（HTML视频控件） 355
creating workers（创建工作线程） 478
cross-browser compatibility of HTML pages（HTML页面的跨浏览器兼容性） 20
detecting geolocation support（检测地理定位支持） 174
detecting support for canvas（检测画布支持）, in code（代码中） 293
detecting support（检测支持）, using Modernizr library（使用Modernizr库） 532
developer tools to manage local storage（开发工具来管理本地存储） 434
exceeding local storage capacity（超出本地存储容量） 458
executing code only after page is fully loaded（只在页面完全加载后执行代码） 64
fallbacks for supported video（支持的视频的"退路"） 362
fitting canvas to window in Fractal Viewer (example)（Fractal Viewer中画布占满窗口（示例）） 517
history of browser storage（浏览器存储的历史） 414~416
loading and displaying HTML documents（加载和显示HTML文档） 14
local storage capacity（本地存储空量） 420
localStorage not working when loading from file（从文件加载时本地存储不能正常工作） 422
methods of determining location（确定位置的方法） 170
mobile devices（移动设备）, canvas support（画布支持） 335
not supporting <canvas>（不支持<canvas>）, displaying text contained in it（显示其中包含的文本） 295
not supporting HTML5 features（不支持HTML5特性）, providing alternative for（提供候选方法） 19
parsing HTML and building DOM from it（解析HTML，并由它构建DOM） 57, 81
running code stored in local storage（运行存储在本地存储中的代码） 104
same origin policy on video（视频的同源策略） 408
security policy（安全策略） 244
storing data using localStorage（使用localStorage存储数据） 108
support for HTML5（对HTML5的支持） 17
support for offline web apps（对离线Web应用的支持） 538
support for Web Workers（对Web工作线程的支持） 482
support for XMLHttpRequest（对XMLHttpRequest的支持）, onload property（onload属性） 239
testing for support of video formats for video loaded by code（测试对由代码加载的视频的视频格式的支持） 368
video encodings supported（所支持的视频编码） 358
video file formats（视频文件格式） 352
video support（视频支持）, determining level of（确定等级） 361, 411
Web Storage support（Web Storage支持） 422
buttons（按钮）
button object（button对象）, onclick property（onclick属性） 154
clearing local storage（清空本地存储） 435
click handlers for video booth（摄像间的点击处理程序）, JavaScript code（JavaScript代码） 377~379
controlling effects in video booth（控制摄像间的效果） 390, 391
CSS styling for video booth（摄像间的CSS样式） 381
handling click event（处理点击事件） 89, 92, 102, 108
HTML for video booth buttons（摄像间按钮的HTML） 375
implementing for video booth（摄像间的实现） 384~386
JavaScript factory code for video booth（摄像间的JavaScript工厂代码） 376
preview button for t-shirt design application（T恤设计应用的预览按钮） 302
selecting between test videos（测试视频间选择） 387
sticky note application（即时贴应用） 431
createSticky handler（createSticky处理程序） 432
toggle or radio buttons（开关或单选按钮） 380
watching position and clearing the watch（监视位置和清除监视） 193
bwcartoon video filter（bwcartoon视频滤波器） 400, 410

C

cache（缓存）, browser（浏览器） 272, 277
cache manifest file for offline web apps（为离线web应用缓存清单文件） 538
callbacks（回调） 254, 277
getting tweets sent from Twitter（获取从Twitter发送的微博） 322
camel case in multi-word variable names（多词变量名的camel case） 42
canPlayType method（canPlayType方法）, video object（video对象） 368~374
"maybe" response（"maybe"响应）, but playback fails（但播放失败） 371
using to determine video format for your browser（用来确定浏览器的视频格式） 369
Canvas API（画布API） 16, 281~348, 540
adding <canvas> element to web page（向Web页面增加 <can-

pin51.com

vas> 元素） 286
background color of canvas（画布的背景色），filling before drawing new squares（画新方块前填充） 306
BE the Browser exercise（扮演浏览器练习）
interpreting call to arc method（解释arc方法调用） 317, 343
browsers not having support for canvas（不提供画布支持的浏览器） 295
call to fillBackgroundColor function（fillBackgroundColor函数调用） 307
<canvas> element vs. SVG graphics（<canvas> 元素与SVG图形） 537
click handler for canvas in Fractal Viewer (example)（Fractal Viewer（示例）中画布的点击处理程序） 515
Code Magnets exercise（代码磁贴练习） 327, 345
Crossword Puzzle（填字游戏） 340, 346
drawImage method（drawImage方法） 333
drawing a smiley face（画笑脸） 321, 344
drawing on the canvas（在画布上绘制） 290~294
circles（圆） 309~317
random circles for t-shirt design app（T恤设计应用的随机圆） 318
using paths to draw shapes with lines（使用路径用线绘制形状） 311
writing drawSquare function to draw squares（编写drawSquare函数画方块） 304
drawing text（绘制文本） 325~332, 345
exercise（练习），drawBird function（drawBird函数） 334, 346
exercise（练习），using path to draw lines and fill shape with color（使用路径画线并用颜色填充形状） 312, 343
fillStyle property of canvas context（画布上下文的fillStyle属性） 308
fitting canvas to browser window in Fractal Viewer (example)（Fractal Viewer（示例）中画布占满浏览器窗口） 517
form for t-shirt application interface（表单实现T恤应用界面） 298
implementing a scratch buffer（实现scratch缓冲区） 395~398
making canvas visible（使画布可见），adding border using CSS（使用CSS增加边框） 288
No Dumb Questions（没有傻问题） 289, 293, 308, 335
Pseudo-Code Magnets exercise（伪代码磁贴练习） 303, 342
reviewing t-shirt design application implementation（回顾T恤设计应用实现） 296
saving and restoring canvas context（保存和恢复画布上下文） 540
and separation of presentation and content（表现和内容分离） 326
Sharpen Your Pencil exercise（动动笔练习）
displaying only new squares in preview（预览中只显示新方块） 306, 342
drawText function（drawText函数） 330
summary of important points（要点总结） 338
text methods and properties（文本方法和属性） 328
translating or rotating canvas（转换或旋转画布） 540
t-shirt design web application（T恤设计Web应用） 282
using as display surface for video（用作视频的显示表面） 408
using <canvas> element for Fractal Viewer (example)（Fractal Viewer（示例）中使用 <canvas>元素） 503, 514
video processing with（视频处理） 392~394
<canvas> element（<canvas>元素）
adding border using CSS（使用CSS增加边框） 288
adding to web page（增加到Web页面） 286
cubicle conversation about the <canvas> element（关于<canvas>元素的交流） 285
partnership with <video> element（与<video>元素的伙伴关系） 339, 388
case sensitivity in JavaScript（JavaScript中区分大小写） 41
cell phone triangulation（蜂窝电话三角定位） 169
chaining（串链）
objects and properties（对象和属性），movie example（电影示例） 141
objects（对象），properties（属性），and method（方法），geolocation（地理定位） 175
character encoding（字符编码），UTF-8 9
character strings（字符串），quoting in JavaScript（JavaScript中加引号） 39
childElementCount property（childElementCount属性），element object（element对象） 158
child elements（子元素）
adding to parent element in the DOM（DOM中增加到父元素） 108
in DOM tree structure（DOM树结构） 100
nth-child selector（nth-child选择器） 548
replaceChild method（replaceChild方法） 270
Chrome 20. 参见 browsers
HTML5 support（HTML5支持） 18
Ogg/Theora video（Ogg/Theora视频） 357
security restrictions on video+canvas operations（视频+画布操作的限制） 371
security restrictions on Web Workers（Web工作线程的安全

限制） 482
.webm video files（.webm视频文件） 352
WebM/VP8 video（WebM/VP8视频） 357
cinema application (example)（影院应用（示例）） 138
adding behavior to Movie object with a method（用方法向Movie对象增加行为） 143~145
creating movie objects（创建movie对象） 139
implementing getNextShowing function（实现getNextShowing函数） 140
Movie constructor function（Movie构造函数） 150, 152
using Movie constructor to create Movie objects（使用Movie构造函数创建Movie对象） 153
using this keyword to reference Movie object（使用this关键字引用Movie对象） 145
circles（圆）, drawing on canvas（画布上绘制） 309~317, 338
arc method（arc方法） 314
converting angle measurement in degrees to radians（度转换为弧度） 317
creating paths（创建路径） 311~313
class attribute（class属性）, <anchor> element（<anchor>元素） 379
class（类）, selecting element by（选择元素） 542
clearInterval method（clearInterval方法） 271
clear method（clear方法）, localStorage object（localStorage对象） 435
clearStorage function (example)（clearStorage函数（示例）） 435
clearWatch method（clearWatch方法） 190
click events（点击事件）
adding handler in canvas application（画布应用中增加处理程序） 302, 347
adding handler in geolocation application（地理定位应用中增加处理程序） 194
adding handlers for sticky notes application（为即时贴应用增加处理程序） 431, 435, 450
assigning handler to element using jQuery（使用jQuery为元素指定处理程序） 534
handler alerting user of button clicks（处理程序提醒用户点击了按钮） 92
handlers for video application（视频应用的处理程序） 376
handling for Add Song button（Add Song按钮的处理） 89
handling for buttons（按钮的处理） 108
close() method（close()方法）, worker object（worker对象） 522, 524
closePath method（closePath方法）, canvas context（画布上下文） 312
codecs
AAC audio（AAC音频） 357
codecs parameter of <source> element's type attribute（<source>元素type属性的codecs参数） 359
defined（定义） 358
H.264 video（H.264视频） 357
main types of（主要类型） 356
Theora video（视频） 357
Vorbis audio（音频） 357
VP8 video（视频） 357
code reuse（代码重用）
functions and（函数） 119
methods and（方法） 146
colors（颜色）
choosing for sticky notes（为即时贴选择）, in stickies application（即时贴应用中） 453~456
fillBackgroundColor function for canvas context（画布上下文的fillBackgroundColor函数） 307
fillRect method（fillRect方法） vs. fillStyle property（fillStyle属性）, canvas context（画布上下文） 308
setting background color of alternating rows in lists（为列表中交替行设置背景色） 548
setting for fillStyle property of canvas context（设置画布上下文的fillStyle属性） 304, 308
specifying in canvas（画布中指定） 338
comma (,)（逗号（,））, separating object properties（分隔对象属性） 132
comments in JavaScript（JavaScript中的注释） 39
computeDistance function (example)（computeDistance函数（示例）） 180
concatenating strings（连接字符串）. 参见 + (plus sign), under Symbols
creating marketing slogans (example)（创建推广口号（示例）） 72
conditionals（条件） 37
testing in while and for loops（while和for循环中测试） 47
while loops（while循环） 46
constructors（构造函数） 146, 160
built-in（内置） 151
creating（创建） 147
LatLong constructor from Google Maps（Google Maps的LatLong构造函数） 183
Map constructor from Google Maps（Google Maps的Map构造函数） 184
Movie constructor function（Movie构造函数） 150, 152

pin5i.com 拼吾爱 最新最全资源分享

using（使用） 148
using Movie constructor to create Movie objects（使用Movie构造函数创建Movie对象） 153
WebSocket 539
containers（容器） 356
defined（定义） 358
MIME type for <source> type attribute（<source> type属性的MIME类型） 359
MP4 container（MP4容器） 357
Ogg container（Ogg容器） 357
in src attribute of <source> element（<source>元素src属性中） 359
WebM container（WebM容器） 357
Content Delivery Network (CDN) companies（内容分发网络（CDN）公司）, encoding services（编码服务） 360
context（上下文）, canvas（画布） 292, 293, 504. 参见 Canvas API
arc method（arc方法） 313～317
beginPath method（beginPath方法） 311, 312
closePath method（closePath方法） 312
drawImage method（drawImage方法） 333
fillRect method（fillRect方法） 292, 304
fillStyle property（fillStyle属性） 330, 331
fillText method（fillText方法） 328, 329, 330, 331
font property（font属性） 329, 330, 331
getting（获取） 302
lineTo method（lineTo方法） 311, 312, 329
moveTo method（moveTo方法） 311, 312, 329
saving and restoring（保存和恢复） 540
stroke method（stroke方法） 329
strokeText method（strokeText方法） 328
textAlign property（textAlign属性） 328, 330, 331
textBaseline property（textBaseline属性） 329
translate and rotate methods（转换和旋转方法） 540
controls attribute（controls属性）, <video> element（<video>元素） 354
cookies 414~416
factors that make them problematic（有问题的方面） 416
Fireside Chat（围炉夜话）, Cookie and Local Storage（Cookie和本地存储） 426
coordinates（坐标）
computing distance between（计算之间距离） 180
latitude and longitude（纬度和经度） 167
coordinates object（coordinates对象） 175, 207
altitude and altitudeAccuracy properties（altitude和altitudeAccuracy属性） 197
latitude and longitude properties（latitude和longitude属性） 173, 175
properties（属性） 190
coords object（coords对象）, latitude and longitude properties（latitude和longitude属性） 173
coords property（coords属性）, position object（position对象） 190
createElement method（createElement方法）, document object（document对象） 99, 157, 335, 450
createSticky function (example)（createSticky函数（示例）） 432
converting to use an array（转换为使用数组） 441
rewriting to store color with sticky note text（重写为随即时贴文本还要存储颜色） 454
stickies application（即时贴应用）, final version（最终版本） 444
createTask function (example)（createTask函数（示例）） 512
cross-document messaging（跨文档传递消息） 543
cross-domain issues with XMLHttpRequest（使用XMLHttpRequest的跨域问题） 243~252
CSS 31
declared standard for styling（指定样式的声明标准） 5
positioning video and canvases（视频和画布定位） 395
property values（属性值） 308
selectors（选择器） 542
styling <canvas> element（<canvas>元素样式）, adding border（增加边框） 288
styling for video booth（摄像间样式） 381
using to set width and height attributes of <canvas>（用来设置<canvas>的width和height属性） 289
using to style sticky notes（用来指定即时贴样式） 429
CSS3 16, 28, 548
page styling（页面样式） 14
curly braces ({ })（大括号（{ }））
enclosing code blocks（包围代码块） 26
enclosing object properties（包围对象属性） 132
currentTime property（currentTime属性）, audio object（audio对象） 533

D

data property（data属性）, event object（event对象） 485, 524
dataTransfer property（dataTransfer属性）, event object（event对象） 543
datatypes（数据类型）
conversions in JavaScript（JavaScript中转换） 41, 45
dynamic typing in JavaScript（JavaScript中动态类型） 39

pin51.com 最新最全的资源分享

primitive types（基本类型） 40
variables in JavaScript（JavaScript中的变量），no strict types（没有严格的类型） 38
Date object（Date对象），getTime method（getTime方法） 140, 272, 442
defining functions（定义函数），with parameters（参数） 120
degradation（降级），graceful（妥善） 19
degrees（度）
angles measured in（用度度量角） 316
converting to radians（转换为弧度） 317
latitude and longitude in（纬度和经度），converting to decimal values（转换为小数值） 167
degreesToRadians function（degreesToRadians函数） 180, 317, 319, 344
deleteSticky function (example)（deleteSticky函数（示例）） 449
event object passed to（传入的event对象），target information（目标信息） 451
deleting object properties（删除对象属性） 135
developer tools built into browsers（浏览器内置的开发工具） 434
direction parameter（direction参数），arc method（arc方法） 315
displayLocation handler function（displayLocation处理函数） 173, 175
altering to show map only once（修改为只显示一次地图） 195
alternative implementation（候选实现） 197, 210
calls from watchPosition（从watchPosition调用），controlling（控制） 206
displaying new marker only after traveling more than 20 meters（只在行走超过20米之后才显示新的标记） 209
distance（距离）
computation and mapping of（计算和建立地图） 197
computing（计算） 180
controlling addition of new map markers（控制新地图标记的增加） 209
writing code to find（编写代码来查找） 181
<doctype> element（<doctype>元素） 3
changes in HTML5（HTML5中的变化） 9, 31
changing HTML 4.01 doctype to HTML5（将HTML 4.01 doctype改为HTML5） 4
omitting（忽略） 9
document object（document对象） 56, 154
createElement method（createElement方法） 99, 101, 335, 450
getElementById method（getElementById方法） 59, 157
getElementsByTagName method（getElementsByTagName方法） 270
properties and method（属性和方法） 157
querySelectorAll method（querySelectorAll方法） 376, 542
querySelector method（querySelector方法） 542
write method（编写方法） 28
documents（文档），cross-document messaging（跨文档消息传递） 543
dollar sign ($)（美元符（$））
$( ) function in jQuery（jQuery中$( )函数） 534
beginning JavaScript variable names（作为JavaScript变量名的开头） 40, 42
domain property（domain属性），document object（document对象） 157
domains（域）
cross-origin issues with XMLHttpRequests（XMLHttpRequest的跨域问题） 244~253
local storage allocated per domain（每个域分配一个本地存储） 422
origin（源），and management of local storage（本地存储管理） 422
DOM (Document object Model)（DOM（文档对象模型） 14, 31, 54~65
adding elements to（增加元素） 100
adding stickies from local storage（从本地存储增加即时贴） 428, 430
creating（创建） 55
creating new <script> elements to continually update data（创建新 <script> 元素持续更新数据） 263, 267
deleting sticky note from（从中删除即时贴） 452
drawing for songs added to playlist（为增加到播放列表的歌曲绘制） 98, 110
empty <ul> element for <li> elements to hold song names（清空<ul>元素用<li>元素包含歌名） 97
getting（获取），creating（创建），adding（增加），or removing elements（或删除元素） 66
getting elements from（从中获取元素），using jQuery（使用jQuery） 534
inability to access or change before page fully loads（页面完全加载之前不能访问或修改） 64
inserting and replacing JSONP <script> elements（插入和替换JSONP <script>元素） 268
interaction of JavaScript with（JavaScript与DOM的交互） 54
No Dumb Questions（没有傻问题） 271

parsing HTML and building DOM from it（解析HTML并由它构建DOM） 81
replaceChild method（replaceChild方法） 270
returning elements by tag name （按标记名返回元素） 270
selecting elements from（从中选择元素）, using Selectors API （使用选择器API） 542
Sharpen Your Pencil exercise（动动笔练习） 61
structure and content of（结构和内容） 56
summary of important points（要点总结） 108
workers not allowed to access（工作线程不允许访问） 480
dot operator (.)（点操作符(.)）
accessing object properties（访问对象属性） 133, 134
invoking methods（调用方法） 151
Drag and Drop API（拖放API） 543
draggable attribute（可拖放属性） 543
drawCircle function (example)（drawCircle函数（示例）） 318
writing（编写） 319
drawImage method（drawImage方法）, canvas context（画布上下文） 333
drawing on the canvas（在画布上绘制） 290~294, 338
arc method（arc方法） 314
drawBird function (example)（drawBird函数（示例）） 334, 346
drawSmileyFace function (example)（drawSmileyFace函数（示例）） 321, 344
drawSquare function (example)（drawSquare函数（示例）） 302, 342
pseudo-code for（伪代码） 303
writing（编写） 304
drawText function (example)（drawText函数（示例）） 327, 330, 345
completing（完成） 331
paths and arcs（路径和弧） 311~318
dropshadows（阴影）, in canvas（画布中） 335
dynamic typing in JavaScript（JavaScript中动态类型） 39

E

effectFunction
calling to apply video filter（调用来应用视频滤波器） 397
used as variable to hold video filter function（用作为变量来包含视频滤波器函数） 391
effects（效果） 410
applying to videos（应用到视频） 389~391
choice of（选择）, video booth（摄像间） 378
creating using canvas context translate and rotate methods（使用画布上下文转换和旋转方法创建） 541
writing special effects for video（为视频编写特效） 399~404
element objects（element对象） 158
returned by getElementById method（getElementById方法返回） 160
elements（元素）
accessing with getElementById（用getElementById访问） 59
adding to the DOM（增加到DOM） 100
creating（创建） 99
getting with getElementById method（用getElementById方法获取） 114, 157
getting with getElementByTagName method（用getElementByTagName方法获取） 154
getting with getElementsByClassName method（用getElementsByClassName方法获取） 154
getting with getElementsByTagName method（用getElementsByTagName方法获取） 269
setting attributes with setAttribute method（用setAttribute方法设置属性） 267
else clauses in if statements（if语句中的else子句） 50
empty strings（空串）
assigning as value to variable（作为值赋给变量） 26
checking for（检查） 95
comparing variables to（比较变量） 108
enableHighAccuracy option （enableHighAccuracy选项）198, 201
encoding your own video（编码你自己的视频） 360
ended event（ended事件）, video（视频） 365
adding event listener for（增加事件监听器） 386
writing handler for（编写处理程序） 367
endedHandler function (example)（endedHandler函数（示例）） 386
enumerating properties of an object（列举一个对象的属性） 133
error handlers（错误处理程序）
for cache errors（缓存错误） 538
Geolocation API（地理定位API） 190, 207
for getCurrentPosition 174, 177~179
for watchPosition 194
video errors（视频错误） 406
in workers（工作线程中） 522
error property（error属性）, video object（video对象） 405
errors（错误）
browsers overlooking small errors in HTML files（浏览器

pin51.com 最新 全 资源 站

忽略HTML文件中的小错误） 9
Geolocation API（地理定位API）
timeout error（超时错误） 200
types of errors（错误类型） 178
handling errors with video playback（处理视频播放的错误） 371
JavaScript syntax（JavaScript语法） 44
localStorage, quota exceeded（配额超限） 458
no XMLHttpRequest errors（没有XMLHttpRequest错误），200 response code（200响应码） 239
video error types（视频错误类型） 405
event handling（事件处理） 89
addEventListener method（addEventListener方法），registering event handler（注册事件处理程序） 367
button click handler（按钮点击处理程序） 102
clearWatch event handler（clearWatch事件处理程序）195
creating handler and assigning it to button onclick property（创建处理程序并赋至按钮onclick属性） 91
handler alerting user that button was clicked（处理程序提醒用户点击了按钮） 92
handleRefresh function（handleRefresh函数） 265
handler for ended video event（ended视频事件的处理程序） 386
handlers for video booth buttons（摄像间按钮的处理程序） 377
handler to make image of canvas drawing（建立画布绘制图像的处理程序） 347
HTTP request handler（HTTP请求处理程序） 221
onclick event handler（onclick事件处理程序）
createSticky (example)（createSticky（示例）） 431
deleteSticky (example)（deleteSticky（示例）） 450
onclick event handler to zoom in on canvas in Fractal Viewer（Fractal Viewer中onclick事件处理程序在画布上放大图像） 515
onload event handler for twitter bird image (example)（onload事件处理程序显示twitter小鸟图像（示例）） 333
onload event handler function for Mighty Gumball (example)（Mighty Gumball的onload事件处理函数（示例）） 229
onload handler as anonymous function（onload处理程序作为匿名函数） 156
onmessage event handler for Web Sockets（Web Sockets的onmessage事件处理程序） 539
onmessage event handler for worker（工作线程的onmessage事件处理程序） 485
onopen event handler for Web Sockets（Web Sockets的onopen事件处理程序） 539
previewHandler function (example)（previewHandler函数（示例）） 302
review of important points（要点回顾） 108
reworking handleButtonClick to obtain song title typed into form by user（修改handleButtonClick来得到用户在表单中键入的歌名） 96
types of events handled by JavaScript（JavaScript处理的事件类型） 95
event object（event对象）
data and target properties（data和target属性） 485
dataTransfer property（dataTransfer属性） 543
target property（target属性） 451
events（事件）
anchor click event（锚点击事件） 376
button click event（按钮点击事件） 90~93
cache（缓存），notification of（通知） 538
canvas click event（画布点击事件） 347, 383, 515
dragstart and dragend events（dragstart和dragend事件） 543
image load event（图像加载事件） 333
properties for event handlers in objects（对象中事件处理程序的属性） 154
request load event（请求加载事件） 221, 222, 229
video（视频） 363
video ended event（视频完成事件） 367, 386
window load event（窗口加载事件） 64, 129, 155, 156, 158, 159
exceptions（异常），QUOTA_EXCEEDED_ERR 458, 468
exercises（练习）
BE the Browser（扮演浏览器） 48, 78
building a DOM（构建DOM） 57, 81
interface element values（接口元素值） 299, 341
interpreting call to arc method（解释arc方法调用） 317, 343
rendering user interface（呈现用户界面） 298
Web Workers（Web工作线程） 488, 526
cliff-hanger（惊险情节），moving to live server（转移到实际服务器） 239, 242
Code Magnets（代码磁贴） 51, 80
canvas（画布），drawText function（drawText函数） 327, 345
geolocation（地理定位） 209
lucky/unlucky web service（幸运/不幸运的Web服务） 223, 224
Movie constructor（Movie构造函数） 150, 152
"what is HTML5?"（"什么是HTML5？"） 30
Crossword Puzzle（填字游戏）
canvas（画布） 340, 346
functions and objects（函数和对象） 161, 163
geolocation（地理定位） 208, 212
HTML5 32, 34
interactions of HTML and JavaScript（HTML与JavaScript的交互） 109, 111

pin5i.com 拼吾爱 最新最全资源分享下载站

JavaScript 76, 84
local storage（本地存储） 465, 471
video（视频） 409, 411
Web apps talking to the Web（Web应用与Web通信） 278, 280
Web Workers（Web工作线程） 525, 528
Don't Try This At Home（不要尝试）
how fast browser can find location（浏览器多快找到位置） 202
local storage（本地存储）, exceeding quota（超出配额） 458, 468
Express Yourself (JavaScript)（表达自己（JavaScript））44
HTML5 archaeology（HTML考古） 20
Pseudo-Code Magnets（伪代码磁贴）, drawSquare function（drawSquare函数） 303, 342
Sharpen Your Pencil（动动笔）
adding song titles to playlist（向播放列表 增加歌名） 65, 82
canvas（画布）, drawText function（drawText函数） 330
canvas（画布）, showing only new squares in preview（预览中只显示新方块） 306, 342
DOM with secret message（有秘密消息的DOM） 61
functions（函数） 122, 162
geolocation（地理定位） 171, 197, 210
HTML5 markup（HTML5标记） 3, 7, 8
JavaScript statements（JavaScript语句） 44, 77
local storage（本地存储）, deleting a sticky（删除即时贴） 447, 448
local storage（本地存储）, problems with stickies implementation（即时贴实现的问题） 437, 467
populating list items from an array（从数组填充列表项） 69, 83
reworking handleButtonClick function（修改handleButtonClick函数） 94, 96
testing for user input on a form（表单上测试用户输入） 94, 96
using setInterval in web applications（web应用中使用setInterval） 266
video control buttons（视频控制按钮）, toggle or radio（开关或单选） 380, 382
video playlist（视频播放列表）, implementing（实现） 364, 365
video（视频）, western and sci-fi effects（西部片和科幻片效果） 400, 410
Web Workers（Web工作线程） 481, 490, 527
Shell Game（果壳游戏）, local storage（本地存储） 425, 466
Who Does What?（连连看）
geolocation options （地理定位选项） 200, 211
HTML5 family of technologies（HTML5技术家族） 16, 33
localStorage API 461, 470
expressions（表达式） 39, 43
evaluating（计算） 44, 77
type conversions（类型转换） 45

F

false (boolean value)（false(布尔值)） 39
family of technologies（技术家族） 12, 29
function of each（各自的功能） 16, 33
file extensions for video（视频文件扩展名） 352, 369
fillBackgroundColor function（fillBackgroundColor函数） 306, 342
calling（调用） 307
Sharpen Your Pencil exercise（动动笔练习） 306, 342
fill method（fill方法）, canvas context（画布上下文） 312
fillRect method（fillRect方法）, canvas context（画布上下文） 292, 304
effects of fillStyle property on（fillStyle属性的效果） 308
fillStyle property（fillStyle属性）, canvas context（画布上下文） 304
closer examination of（进一步分析） 308
fillText method（fillText方法）, canvas context（画布上下文） 325, 328
using with tweet text (example)（使用微博文本（示例）） 331
film noir video filter（黑白片视频滤波器） 374, 399
Firefox. 参见 browsers
HTML5 support（HTML5支持） 18
Ogg/Theora video（Ogg/Theora视频） 357
.ogv video files（.ogv视频文件） 352
WebM/VP8 video（WebM/VP8视频） 357
Fireside Chats（围炉夜话）
Cookie and Local Storage（Cookie和本地存储） 426
XMLHttpRequest and JSONP（XMLHttpRequest和JSONP） 260
firstChild property（firstChild属性）, element object（element对象） 158
Flash
HTML5 versus 284
use to solve cross-browser issues（用来解决跨浏览器问题） 20
Flash Video（Flash视频） 358
flexbox layout（flexbox布局） 548
floating point numbers（浮点数）
conversion of integers to in expressions（表达式中整数转换为） 45

pin51.com 最新最全电子书资源分享网站

storing in local storage（本地存储中存储） 423

font property（font属性），canvas context（画布上下文） 329
  setting for tweet text (example)（设置微博文本（示例）） 331

<footer> element（<footer>元素） 546

for loops（for循环） 47
  deciding between while loops and（决定while循环还是） 47
  evaluating (example)（计算（示例）） 48
  if/else statements in（if/else语句） 51

forms（表单） 16, 85~112
  adding button to（增加按钮） 91
  adding t-shirt design form to HTML page（向HTML页面增加T恤设计表单） 301
  adding tweets to <select> element in form（向表单中的<select>元素增加微博） 323
  checking whether user entered text input（检查用户是否输入文本） 96
  client-side（客户端），accessing values in（访问值） 296
  displaying playlist on HTML page（在HTML页面上显示播放列表） 97
  getting text from input element（从输入元素获取文本） 94, 108
  HTML5 document to hold form and list element for playlist（HTML5文档包含播放列表的表单和列表元素） 87
  playlist manager application（播放列表管理器应用） 102
  sticky note application（即时贴应用） 429
    updating to add colors（更新来增加颜色） 453
  tracking position（跟踪位置） 193
  t-shirt application interface（T恤应用用界面） 298
  using JavaScript for real interactivity（使用JavaScript实现真正的交互性） 23

forward slashes (//)（前斜线（//）），beginning JavaScript comments（作为JavaScript注释的开头） 39

Fractal Explorer application（Fractal Explorer应用），building (example)（构建（示例）） 494, 503
  creating Fractal Viewer HTML page（创建Fractal Viewer HTML页面） 503
  creating workers（创建工作线程） and giving tasks to（分配任务） 508
  final test drive（最后的测试） 519
  getting workers started（启动工作线程） 510
  handling click events to zoom in（处理点击事件来放大） 515
  how number of workers affects performance（工作线程个数如何影响性能） 520
  implementing workers（实现工作线程） 511
  managing fractal generations（管理分形代） 518
  processing workers' results（处理工作线程的结果） 514
  ready-baked code for Mandelbrot Set computation（Mandelbrot集计算的成品代码） 504~507
  tasks（任务） 512
  writing the code（编写代码） 509

fractal image（分形图像），Mandelbrot Set as（Mandelbrot集） 495

FTP programs（FTP程序） 232

fullscreen playback of video（视频的全屏播放） 360

functions（函数） 113~130, 160
  anatomy of（剖析） 121
  assigning to window object（指定到window对象），onload property（onload属性） 75
  built in（内置） 119
  callbacks（回调） 254
  constructor（构造函数） 147
  creating your own（创建你自己的） 115
  Crossword Puzzle（填字游戏） 161, 163
  declarations（声明），placement of（放置） 127
  defining（定义） 71
  how it works（如何工作） 116
  inability to pass to Worker constructor（不能传递到工作线程构造函数） 491
  interview with（访谈） 119
  invoking（调用） 116
  life span of variables（变量生命期） 125
  Math library（Math库） 75
  methods versus（方法与） 151
  naming（命名） 121
  No Dumb Questions（没有傻问题） 121, 127
  object passed to（对象传递到），accessing properties of（访问属性） 134
  parameters and arguments（形参和实参） 120
  passing a function to a function（将函数传递到函数） 175
  passing arguments to parameters（将行参传递实参） 122, 162
  passing objects to （传递对象） 134, 136
  return statements in the body（体中的返回语句） 117
  reworking as methods（修改为方法） 143
  scope of local and global variables（局部和全局变量的作用域） 124
  setting special effects for videos（设置视频的特效） 391

functions（函数）
  Sharpen Your Pencil exercise（动动笔练习） 122, 162
  using as values（用作为值） 129
  as values（作为值） 128

variables defined in（其中定义的变量） 123, 160

G

generation（代）, fractal (example)（分形（示例）） 518
geolocation（地理定位） 165~212
accuracy of location（位置精度） 191
adding a Google marker to your map（为地图增加Google标记） 186
adding a map to your page（向页面增加地图） 183
Crossword Puzzle（填字游戏） 208, 212
displaying map on your page（在页面上显示地图） 184
Don't Try This At Home exercise（不要尝试练习） 202
error handler（错误处理程序） 177
finding how fast browser can find location（查看浏览器多快找到位置） 202
getCurrentPosition method（getCurrentPosition方法） 175
how getCurrentPosition works（getCurrentPosition如何工作） 176
mapping your position（映射位置） 182
No Dumb Questions（没有傻问题） 166, 197
other uses of Google Maps（Google Maps的其他用法） 188
server required to test code on mobile devices（需要服务器在移动设备上测试代码） 179
Sharpen Your Pencil exercise（动动笔练习） 171
alternative implementation for displayLocation（displayLocation的候选实现） 197, 210
specifying options（指定选项） 201
success handler for getCurrentPosition（getCurrentPosition的成功处理程序） 174
summary of important points（要点总结） 207
timeout and maximumAge options（timeout和maximum-Age 选项） 199
tracking movements（跟踪移动） 192
clearWatch handler（clearWatch处理程序） 195
with markers on a map（地图上用标记） 204
watchLocation handler（watchLocation处理程序） 194
watchPosition method（watchPosition方法） 192
Who Does What? exercise（连连看练习） 200, 211
Geolocation API（地理定位API） 16. 参见geolocation（地理定位）
components of（组件） 190
getCurrentPosition method（getCurrentPosition方法） 174, 177, 190, 207
interview with（访谈） 189
position options（position选项） 198
watchPosition method（watchPosition方法） 194
geolocation property（geolocation属性）, navigator object（navigator对象） 174
getAttribute method（getAttribute方法）, element object（element对象） 158, 379
getContext method（getContext方法）, canvas object（canvas对象） 292, 293
getCurrentPosition method（getCurrentPosition方法）, geolocation object（对象） 174, 190, 207
error handler for（错误处理程序） 177
how it works（如何工作） 176
getElementById method（getElementById方法）, document object（document对象） 58, 72, 157
using to locate element and change its content（用来定位元素并改变内容） 59, 60
getElementsByClassName method（getElementsByClassName方法）, document object（document对象） 157
getElementsByTagName method（getElementsByTagName方法）, document object（document对象） 157, 270
getFormatExtension function (example)（getFormatExtension函数（示例）） 369, 370
adding to video booth code（增加到摄像间代码） 383
getItem method（getItem方法）, localStorage object（localStorage对象） 419, 421
getMyLocation function (example)（getMyLocation函数（示例）） 172
getNextShowing function (example)（getNextShowing函数（示例）） 140
GET request (HTTP)（GET请求（HTTP）） 220
getStickiesArray function (example)（getStickiesArray函数（示例）） 443
getTimeFromString function (example)（getTimeFromString函数（示例）） 140
getTime method（getTime方法）, Date object（Date对象） 140, 272, 442
global variables（全局变量） 123, 160
life cycle of（生命期） 125
overuse in JavaScript（JavaScript中滥用） 127
reasons for sparing use of（减少使用的原因） 127
shadowing（屏蔽） 126
Google Chrome 参见 Chrome
Google Maps 182
adding marker to your map（向地图增加标记） 186
LatLong constructor（LatLong构造函数） 183
other uses of（其他用法） 188
GPS (Global Positioning System)（GPS（全球定位系统）） 168

devices without（没有GPS的设备）, using Geolocation API on（使用地理定位API） 189
graceful degradation（妥善降级） 19
graphics（图形）, SVG 537
Greenwich（格林威治）, England（英国）, measured from（经度测量） 167

H

H.264 video format（H.264视频格式） 352, 356, 357
handleButtonClick function (example)（handleButtonClick函数（示例）） 90
  assigning to button onclick property（赋给按钮onclick属性） 91
  code to create <li> child element and add it to DOM（创建<li>子元素并增加到DOM的代码） 101
  reworking to obtain song title typed into form by user（修改来得到用户在表单中键入的歌名） 94, 96
  running when user clicks button（用户点击按钮时运行） 93
handleClick function (example)（handleClick函数（示例）） 515, 516
handleControl function (example)（handleControl函数（示例）） 377, 384
  implementing rest of video controls（实现其余的视频控件） 385
handleRefresh function (example)（handleRefresh函数（示例）） 265, 267, 272
  adding lastreporttime parameter（增加lastreporttime参数） 275
handleRequest function (example)（handleRequest函数（示例）） 523
<head> element（<head>元素）
  <link> and <script> elements（<link>和<script>元素） in 5
  placing <script> elements in（放置<script>元素） 53
  replacing <script> child elements（替换<script>子元素） 270
<header> element（<header>元素） 546
heading property（heading属性）, coordinates object（coordinates对象） 190, 197
<hgroup> element（<hgroup>元素） 546
hosting services（托管服务） 230, 232
HTML
  interaction of JavaScript with markup 54
  parsing and building DOM from 57
HTML5
  converting HTML 4.01 document to（转换 HTML 4.01为） 2~5
  Crossword Puzzle（填字游戏） 32, 34
  interactions of HTML and JavaScript（HTML与JavaScript交互） 109, 111
  family of technologies（技术家族） 12, 16, 33
  final recommendation of standard（最终推荐标准） 20
  handling older browsers（处理较老的浏览器） 19
  how it really works（到底如何工作） 14
  improvements in markup（标记改进） 14
  interview with（访谈） 11
  JavaScript as integral part of（JavaScript作为一部分） 21, 118, 130
  JSON and JSONP（JSON和JSONP） 271
  magnets exercise（磁贴练习）, "what is HTML5?"（什么是"HTML5"） 30
  markup（标记）, JavaScript APIs, and CSS 29
  Mighty Gumball application page (example)（Mighty Gumball应用页面（示例）） 218
  new capabilities and features（新功能和特性） 12
  new elements（新元素）, reference（引用） 545
  No Dumb Questions（没有傻问题） 9, 20, 28, 284
  page for t-shirt design application（T恤设计应用页面） 300
  prerequisites for learning（学习的先决条件） 10
  Sharpen Your Pencil exercise（动动笔练习） 3, 7
  summary of important points（要点总结） 31
  support in browsers（浏览器的支持） 18
  versus using Flash or custom applications（与使用Flash或定制应用相比） 284
  what is it（是什么） 12
  what you can do with HTML5 and JavaScript（利用HTML5 JavaScript你能做什么） 22
  Who Does What? exercise（连连看练习） 16, 33
HTML entities in tweets on canvas（画布上微博中的HTML实体） 335
HTTP-based request/response model（基于HTTP的请求/响应模型） 539
HTTP-based video streaming（基于HTTP的流式视频） 404
HTTP used with XMLHttpRequest（HTTP结合XMLHttpRequest）
  HTTP requests（HTTP请求） 219, 220, 239
  HTTP responses（HTTP响应） 219, 221, 239
    accessing returned data（访问返回的数据） 222
  server required to use（使用时需要有服务器） 230

I

id attribute（id属性），参见 getElementById method（getElementById方法），document object（document对象）
accessing elements by（访问元素） 59
<canvas> element（<canvas>元素） 290
needed to delete sticky note (example)（删除即时贴所需（示例）） 450
<script> element（<script>元素） 267
<video> element（<video>元素） 353
IE. *See* Internet Explorer
if statements（if语句） 49
if/else statements（if/else语句） 50
<iframe> element（<iframe>元素） 543
IIS servers（IIS服务器），configuring MIME types（配置MIME类型） 371
image（图像），making of t-shirt design drawn in canvas（画布中绘制T恤设计） 347
image objects（image对象）
creating（创建） 333
Image constructor（Image构造函数） 335
<img> element（<img>元素），<canvas> versus 285
iMovie, encoding video with（编码视频） 360
importScripts global function（importScripts全局函数），Web Workers（Web工作线程） 493, 511
using to make JSONP requests（用来建立JSONP请求） 523
Indexed Database API 543
indexes（索引），array（数组） 75
InfoWindow object（InfoWindow对象） 187
init function（init函数） 64
as anonymous function（作为匿名函数） 159
inline code（内联代码），writing in HTML5 <script> element（编写HTML5<script>元素） 5
innerHTML property（innerHTML属性） 60
element object（element对象） 158
using to change element content（用来改变元素内容） 62
insertBefore method（insertBefore方法），element object（element对象） 158
integers（整数）
conversions to floating point numbers in expressions（表达式中转换为浮点数） 45
converting from strings（从字符串转换） 423
storing in localStorage as strings（localStorage中作为字符串存储） 423
interface transformations on elements（元素上接口转换），using jQuery（使用jQuery） 535
Internet Explorer. 参见 browsers（浏览器）
canvas support（画布支持），versions 9 and later（IE9及以后版本） 294
HTML5 support（HTML5支持） 18
MP4/H.264 video（MP4/H.264视频），supported by（支持） IE9 357
no Web Worker support prior to IE10（IE10之前不支持Web工作线程） 482
versions 6 and 7（版本6和7），not supporting localStorage（不支持localStorage） 104
video file format（视频文件格式） 352
XMLHttpRequest object and（XMLHttpRequest对象） 240
interval timer（间隔定时器），stopping（停止） 271
interviews（访问）
with function（函数） 119
with Geolocation（地理定位） 189
with HTML5 11
with JavaScript（JavaScript） 24, 477
with Video（视频） 388
with XMLHttpRequest 225, 240
invoking functions（调用函数） 116
with arguments（提供参数） 120
IP address（IP地址），location information based on（基于位置信息） 168
isButtonPushed helper function (example)（isButtonPushed辅助函数（示例）） 379, 384
iterating through localStorage（通过localStorage迭代） 424

J

JavaScript 31, 35~54
adding behavior with（增加行为） 35
adding to web pages（增加到Web页面） 53
APIs 15
arrays（数组） 67, 69, 71~73
and objects（对象） 133
passing to functions（传递到函数） 122
passing to Web Worker（传递到Web工作线程） 484
returned from getElementsByTagName（从getElementsByTagName返回） 270, 271
returned from querySelectorAll（从querySelectorAll返回） 376
and <select> element options（<select> 元素选项） 303
storing in localStorage（localStorage中存储） 439, 445

associative arrays（关联数组） 424
BE the Browser exercise（扮演浏览器练习） 48, 78, 81
browser security policy and（浏览器安全策略） 244~246
Code Magnets exercise（代码磁贴练习） 51, 80
canvas（画布），drawText function（drawText函数） 327
displayLocation handler function（displayLocation处理函数） 209
Movie constructor（Movie构造函数） 150, 152
creating dynamic HTML page content（创建动态HTML页面内容） 28
Crossword Puzzle（填字游戏） 76, 84
functions and objects（函数和对象） 161, 163
interactions with HTML（与HTML交互） 109, 111
declaring a variable（声明变量） 38~40
default scripting language in HTML5（HTML5中默认脚本语言） 5
drawing on canvas（画布上绘制） 285
enabling preview button on t-shirt design app（T恤设计应用中启用预览按钮） 302
expressions（表达式） 43
functions（函数） 113~130, 162
summary of important points（要点总结） 160
getElementById 58
handling events（处理事件） 89
review of important points（要点回顾） 108
how it handles tasks of typical page（如何处理典型页面的任务） 474
how it works（如何工作） 36
and HTML5 21, 22, 118, 130
Image constructor（Image构造函数） 335
including additional files in worker（工作线程中包含额外文件） 493
interaction with page through DOM（通过DOM与页面交互） 15, 58
interaction with your page（与页面交互） 54
interview with（访谈） 24, 477
jQuery 534
line-by-line analysis of code（代码的逐行分析） 26
making decisions（决策），using conditional statements（使用条件语句） 49
making HTTP requests from（HTTP请求） 220~225
making use of HTML5 family of technologies（利用HTML5技术家族） 24
Modernizr library（Modernizr库） 532
No Dumb Questions（没有傻问题） 28, 41, 47, 73
events and handlers（事件和处理程序） 95
functions（函数） 121, 127
functions and objects（函数和对象） 151
objects（对象） 158
Web Workers（Web工作线程） 491
objects（对象） 113, 131~161
summary of important points（要点总结） 160
property values in（属性值） 308
repetitive tasks using loops（使用循环的重复任务） 46~48
reserved words（保留字） 41
Sharpen Your Pencil exercise（动动笔练习）
displayLocation implementation（displayLocation实现） 197, 210
functions（函数） 122, 162
populating list items from an array（由数组填充列表项） 69, 83
populating playlist items using an array（使用数组填充播放列表项） 65, 82
reworking handleButtonClick function（修改handleButtonClick函数） 94
statements（语句） 44, 77
using setInterval in Web applications（Web应用中使用setInterval） 266
single-threaded model（单线程模型） 474, 477
summary of important points（要点总结） 75
syntax（语法） 39
testing code in HTML page（测试HTML页面代码） 27
using with HTML5（使用HTML5） 22
working with canvas and video（处理画布和视频） 388
writing（编写） 25
jQuery 534
online documentation and tutorials（联机文档和教程） 535
JSON (JavaScript object Notation)（JSONN（JavaScript对象记法）） 226
adding support to web application（增加对web应用的支持） 236
converting movie object to and from JSON string format (example)（movie对象与JSON字符串格式之间的转换（示例）） 227
creating string representation of an array（创建数组有字符串表示） 441, 467
Crossword Puzzle（填字游戏） 278, 280
as data（作为数据） 249~251
gumball sales returned from Mighty Gumball (example)（Mighty Gumball返回的糖果销售信息（示例）） 233
HTML5 and 271
and JSONP 252
No Dumb Questions（没有傻问题） 271
performance issues with use to convert to and from strings（用来与字符串之间转换存在的性能问题） 442

pin51.com 拼吾爱 最新最全的资源分享站

tweets returned from Twitter (example)（Twitter返回的微博（示例）） 323
URL to include last report time (example)（URL包含最后一次报告时间（示例）） 275
XML and 226
and XMLHttpRequest 225
JSON.parse method（JSON.parse方法） 226
converting JSON string back to object（JSON字符串与对象之间的转换） 227
using on arrays or objects retrieved from localStorage（从localStorage获取数组或对象） 443, 445
using when object stored in localStorage（localStorage中存储对象时使用） 455
JSONP (JSON with Padding)（JSONP（带填充的JSON）） 240, 247
Crossword Puzzle（填字游戏） 278, 280
Fireside Chat with XMLHttpRequest（与XMLHttpRequest围炉夜话） 260
HTML5 and 271
introduction to（介绍） 252
making call to Mighty Gumball JSONP API (example)（Mighty Gumball JSONP API调用（示例）） 257
making call to Twitter JSONP API (example)（Twitter JSONP API调用（示例）） 322
making it dynamic（使它为动态） 264~271
No Dumb Questions（没有傻问题） 271
P in JSONP（JSONP中的P）, defined（定义） 253
security and（安全性） 259
summary of important points（要点总结） 277
updating web application with（更新web应用） 256~263
using importScripts to make requests（使用importScripts建立请求） 523
JSON.stringify method（JSON.stringify方法） 226
converting object to JSON string format（对象转换为JSON字符串格式） 227
storing object in local storage（对象存储在本地存储） 454
using to store arrays or object in localStorage（用来在localStorage中存储数组或对象） 442, 445

K

key method（key方法）, localStorage object（localStorage对象） 424, 430
key/value pairs（键/值对）
in browser's local storage（浏览器本地存储中） 417
creating unique keys（创建唯一键） 442
managing keys in stickies application（即时贴应用中管理键） 433
passing key each time sticky note is added to DOM（每次即时贴增加到DOM时传递键） 450
storing in an array（存储在数组中） 439
in string form（字符串形式）, getting and setting in local storage（本地存储中获取和设置） 419, 421
uniqueness of keys in local storage（本地存储中键的唯一性） 422
using key to remove item from localStorage and array（从localStorage和数组使用键删除项） 451

L

<label> element（<label>元素） 453
lastreporttime query parameter (example)（lastreporttime查询参数（示例）） 275
latitude and longitude（纬度和经度） 167
accuracy of geolocation information（地理定位信息的精度） 179
latitude and longitude properties（latitude和longitude属性）, coordinates object（coordinates对象） 184
layouts（布局）, new（新增）, in CSS3（CSS3中） 548
length property（length属性）
arrays（数组） 68
localStorage object（localStorage对象） 424, 430, 432
letter-boxing video（letter-boxing视频） 354
line breaks in HTML（HTML中的换行） 26
lines（线）, drawing shapes in canvas（画布中绘制形状） 311
lineTo method（lineTo方法）, canvas context（画布上下文） 311
lineWidth property（lineWidth属性）, canvas context（画布上下文） 312
<link> elements（<link> 元素）, within <head> element（<head>元素）, pointing to CSS stylesheet（指向CSS样式表） 5
Linux
Apache server（Apache服务器）, configuring MIME types（配置MIME类型） 371
setting up server on（建立服务器） 231
lists（列表）
adding songs to playlist with JavaScript (example)（用JavaScript向播放列表增加歌曲（示例）） 65, 82
creating <li> elements（创建 元素） 99
filling in items using array (example)（使用数组填充数据项（示例）） 69
finding all child <li> elements of element with id of play-

list（查找id为playlist的元素的所有子<li>元素），using jQuery（使用jQuery） 535
playlist manager (example)（播放列表管理器应用（示例））
adding <li> child element to <ul> parent（向<ul>父元素增加<li>元素） 100, 110
 elements to hold song names（<li>元素包含歌名） 97, 99
 element to hold playlist（<ul>元素包含播放列表） 87, 97
setting background color for alternating rows（设置隔行背景色） 548
stickies application (example)（即时贴应用（示例））
creating <li> element to hold sticky note（创建<li>元素包含即时贴） 430
stickies from localStorage inserted into <ul> element（localStorage中的即时贴插入到<ul>元素） 430
 element to hold stickies（<ul>元素包含即时贴） 429
load event（加载事件）
and image onload property（图像onload属性） 333
and window.onload property（window.onload属性） 64
load method（load方法）
audio object（audio对象） 533
video object（video对象） 385
Local Files origin（本地文件源） 422
local storage（本地存储） 16, 108, 413~472
5MB limit and domain（5MB限制和域） 422
access by workers（工作线程访问） 491
array-based code（基于数组的代码）, integrating into stickies application（集成到即时贴应用） 443
associative arrays（关联数组） 424
browser-based（基于浏览器）, instead of cookies（而不是cookie） 23
browser issues with file://（file://的浏览器问题） 422
browsers' tools for managing（浏览器工具用于管理） 434
browser storage（浏览器存储）, history of（历史） 414~416
code to save playlist（保存播放列表的代码） 104
Crossword Puzzle（填字游戏） 465, 471
deleting items（删除数据项） 446
designing your application storage（设计你的应用存储） 457
Don't Try This At Home exercise（不要尝试练习） 458, 468
exceeding capacity of（超出容量） 458
Fireside Chat（围炉夜话）, Cookie and Local Storage（Cookie和本地存储） 426
how HTML5 Web storage works（HTML5 Web存储如何工作） 417
how local storage API works（本地存储API如何工作） 420
IndexedDB and Web SQL（IndexedDB和Web SQL） 543
naming of keys（键命名） 433, 445
No Dumb Questions（没有傻问题） 422, 425, 433, 442, 445
problems with using length to store keys（使用长度存储键的问题） 436
sessionStorage object（sessionStorage对象） 460
Sharpen Your Pencil exercise（动动笔练习）
deleting a sticky（删除即时贴） 447, 448
problems with stickies implementation（即时贴实现的问题） 437, 467
Shell Game exercise（果壳练习） 425, 466
stickies application（即时贴应用） 418, 428
storing arrays（存储数组） 440
storing non-String data types（存储非字符串数据类型） 439
storing numbers（存储数字） 423
storing objects（存储对象） 454~457
summary of important points（要点总结） 464
using（使用） 462
Who Does What? exercise（连连看练习） 461, 470
localStorage object（localStorage对象） 418
clear method（clear方法） 435
getItem method（getItem方法） 419, 421
key method（key方法） 424
length property（length属性） 424
removeItem method（removeItem方法） 433, 446, 449
setItem method（setItem方法） 418, 421
treating as associative array（处理为关联数组） 424
localStorage property（localStorage属性）, Window object（Window对象） 422
local variables（本地变量） 123, 160
life cycle of（生命期） 125
shadowing global variable（遮蔽全局变量） 126
location（位置）
accuracy of（精度） 191
Geolocation API in JavaScript（JavaScript中地理定位API） 166
how Geolocation API determines it（地理定位API如何确定） 168
location aware（位置感知） 165
loop attribute（loop属性）, <video> element（<video>元素） 354
loop property（loop属性）, video object（video对象） 385

looping（循环） 37, 46~48
deciding between while and for loops（while和for循环中选择） 47
evaluating while and for loops (example)（评价while和for循环（示例）） 48
for loops（for循环） 47
using arrays with loops（结合循环使用数组） 69, 75
while loops（while循环） 46

M

Mac
Apache server（Apache服务器）, configuring MIME types（配置MIME类型） 371
setting up server on（建立服务器） 231
task monitor on OS X（OS X上任务监视器） 520
makeImage function (example)（makeImage函数（示例）） 347
makeServerRequest function (example)（makeServerRequest函数（示例）） 523
Mandelbrot, Benoit 495
Mandelbrot Set. 参见 Fractal Explorer application（Fractal Explorer应用）, building（构建）
computing（计算） 496
equation（公式） 495
explorer for（探险家） 494
ready-baked code for computing（计算的成品代码） 504~507
using multiple workers to compute（使用多个工作线程来计算） 497~500
mapOptions object（mapOptions对象） 184
mapping your position（映射位置） 182
maps（地图）
adding markers to（增加标记） 186, 204
adding to a page（增加到页面） 183
displaying on your page（页面上显示） 184
testing map display on your page（页面上测试地图显示） 185
<mark> element（<mark>元素） 547
markers（标记）, adding to map（增加到地图） 186, 204
controlling frequency of new markers（控制新标记的频度） 209
optimizing marker usage（优化标记使用） 206
markup（标记）, new（新增） 16, 533
Math.floor function（Math.floor函数） 70, 304, 319
Math library（Math库） 73, 75
Math.PI 317
Math.random function（Math.random函数） 70, 304, 319
x, y, and width of squares drawn on canvas（画布上绘制的方块的x, y和宽度） 308
maximumAge option（maximumAge选项） 199, 201
message handler（消息处理程序）, writing for worker（为工作线程编写） 486
messages（消息）
data that can be sent（可以发送的数据） 484, 491
receiving by Web Worker（web工作线程接收） 486
receiving from Web Workers（从Web工作线程接收） 485
sending and receiving using Web Sockets（使用Web Sockets发送和接收） 539
sending from Web Worker（从Web工作线程发送） 486
sending to Web Workers（发送到Web工作线程） 484
messaging（消息传递）, cross-document（跨文档） 543
<meta> tags（<meta>标记） 31
omitting（忽略） 9
specifying in HTML5（HTML5中指定） 4
methods（方法） 142, 160
code reuse and（代码重用） 146
converting functions to（转换函数为） 143
functions versus（函数） 151
this keyword（this关键字）, how it works（如何工作） 149
Microsoft. 参见 Internet Explorer; Windows systems
Smooth Streaming（平滑流式机制） 404
Web Platform Installer（Web平台安装程序） 231
Mighty Gumball application (example)（Mighty Gumball应用（示例）） 214~218
browser cache（浏览器缓存）, watching out for（监视） 272
displaying sales（显示销售情况） 230
improving the display（改进显示） 235
making JSONP dynamic（使JSONP动态） 264~271
moving to live server（转移到实际服务器） 237~246
options to circumvent cross-origin request problems（克服跨源请求问题的选项） 247~251
removing duplicate sales reports（删除重复的销售报告） 273
reviewing the specs（查看规范） 228
reworking code to use JSON（修改代码来使用JSON） 236
testing locally（本地测试） 230, 234
updating code to use JSONP（更新代码来使用JSONP） 256~263
updating JSON URL with lastreporttime（用lastreporttime更新JSON URL） 275

writing onload handler function（编写onload处理函数） 229
milliseconds since 1970（自1970年以来的毫秒数） 442
MIME types（MIME类型）
application/xhtml+xml 536
making sure server is serving video files with correct type（确保服务器提供正确类型的视频文件） 371
of video files（视频文件） 359, 369
mobile browsers（移动浏览器） 20
HTML5 support（HTML5支持） 18
mobile devices（移动设备）
browser support for offline Web apps（离线Web应用的浏览器支持） 538
canvas on（画布） 335
testing geolocation code（测试地理定位代码） 179
Modernizr library（Modernizr库）, JavaScript 532
movements（移动）, tracking（跟踪） 192~198
form to start and stop tracking（开始和停止跟踪的表单） 193
moveTo method（moveTo方法）, canvas context（画布上下文） 311
.mp3 audio（.mp3音频） 533
.mp4 video files（.mp4视频文件） 352
MP4 container（MP4容器） 357
MPEG-LA group（MPEG-LA组） 357
multi-core processors（多核处理器） 500
muted property（muted属性）, video object（video对象） 385

N

names（名）
of functions（函数） 121
guide to better naming of variables（更好的变量命名指南） 42
local and global variables with same name（同名的局部和全局变量） 126
localStorage keys（localStorage键） 433, 445
of variables（变量的） 40
namespaces（命名空间）, XHTML 536
<nav> element（<nav>元素） 547
navigator object（navigator对象）, geolocation property（geolocation属性） 174
new keyword（new关键字）, using with constructors（结合构造函数使用） 148, 160
nextVideo handler function（nextVideo处理函数） 367
No Dumb Questions（没有傻问题）
canvas（画布） 289, 293, 308
events and handlers（事件和处理程序） 95
functions（函数） 121, 127
functions and objects（函数和对象） 151
geolocation（地理定位） 166, 197
HTML5 9, 20
HTML5 web applications（HTML5 Web应用） 284
JavaScript 41, 47, 73
JavaScript and HTML5 technologies（JavaScript和HTML5技术） 28
local storage（本地存储） 422, 425, 433, 442, 445
objects（对象） 158
talking to the Web（与Web通信） 271
video（视频） 360, 371
Web Workers（Web工作线程） 491
nth-child selector（nth-child选择器） 548
numbers（数字）
conversions to other types in expressions（表达式中转换为其他类型） 45
primitive type in JavaScript（JavaScript中的基本类型） 40
storing in local storage（本地存储中存储） 423
numeric expressions（数值表达式） 43

O

<object> element（<object>元素）, using inside <video> element（<video>元素中使用） 362
object literals（对象直接量） 151
objects（对象） 40, 113, 131~161
adding or deleting properties at any time（任何时间增加或删除属性） 135
array（数组） 73
arrays of（数组） 457
in the browser（浏览器中） 154
built-in versus created by users（内置与用户创建） 159
constructors（构造函数） 147
converting to and from JSON string format（与JSON字符串格式之间的转换） 226, 227
creating（创建） 132
movie object (example)（movie对象（示例）） 138
using constructors（使用构造函数） 148, 153
Crossword Puzzle（填字游戏） 161
methods（方法） 142
No Dumb Questions（没有傻问题） 151, 158
passing to functions（传入函数） 136
properties（属性） 132
storing in localStorage（存储在localStorage中） 445, 454

storing shapes drawn in canvas（存储画布中绘制的形状）as 336
summary of important points（要点总结） 160
this keyword（this关键字） 144
uses of（使用） 133
writing versus creating with a constructor（用构造函数编写和创建） 151
offline web applications（离线web应用） 16, 538
Ogg container format（Ogg容器格式） 357
Ogg/Theora video encoding（Ogg/Theora视频编码） 356, 357
Ogg/Vorbis audio encoding（Ogg/Vorbis音频编码） 356, 357, 533
.ogv video files（.ogv视频文件） 357
onclick property（onclick属性）, button objects（button对象） 91, 154
adding event handler function to（增加事件处理函数） 91, 450
onerror handler（onerror处理程序）, using in workers（工作线程中使用） 522
onload handler function（onload处理函数） 64, 229
and anonymous functions（匿名函数） 129, 156
using to load page before accessing the DOM（访问DOM之前用来加载页面） 64
writing with jQuery（用jQuery写） 534
onload property（onload属性）
image object（image对象） 333
window object（window对象） 156
assigning function to（指定函数） 64, 75, 129, 156, 265
XMLHttpRequest object（XMLHttpRequest对象） 239
browsers not supporting（浏览器不支持）, work around for（避开） 241
onmessage event handler（onmessage事件处理程序） 485
opacity property（opacity属性） 548
opacity（透明度）, transitioning from opaque to translucent（从不透明到半透明） 548
open event（open事件）, Web Sockets 539
Opera. 参见 browsers（浏览器）
HTML5 support（HTML5支持） 18
not supporting XMLHttpRequest Level 1（不支持XMLHttpRequest Level 1） 241
Ogg/Theora video（Ogg/Theora视频） 357
.ogv video files（.ogv视频文件） 352
WebM/VP8 video（WebM/VP8视频） 357
<option> element（<option>元素）, in stickies application（即时贴应用中） form 453
options（选项）, Geolocation API（地理定位API） 198, 201
summary of（总结） 207
Who Does What exercise（连连看练习） 200, 211
overlays（重叠）, Google Maps 188

P

palindromes（回文） 51
panTo method（panTo方法）, map object（map对象） 204
parameters（行参）, function（函数） 120
names of（名） 121
passing arguments to parameters（向行参传入实参） 116, 122, 162
parent element（parent元素）
adding a child element with appendChild（用appendChild增加子元素） 100~102
adding child elements to（增加子元素） 108
in DOM（DOM中） 100
parseFloat function（parseFloat函数） 423
parseInt function（parseInt函数） 423
parse method（parse方法）. *See* JSON.parse method（JSON.parse方法）
passing by value（按值传递） 136
passing an object reference to a function（向函数传入对象引用） 136
paths and arcs in canvas（画布中的路径和弧） 310, 338
arc method（arc方法） 313~315
drawing a smiley face（画笑脸） 344
using arc method and path（使用arc方法和路径） 343
using arc method and draw a circle（使用arc方法画圆） 316
using arc method and trace a path（使用arc方法跟踪路径） 316
using paths to draw shapes（使用路径绘制形状） 311
pause method（pause方法）
audio object（audio对象） 533
video object（video对象） 385
PC, setting up server on（建立服务器） 231
Phrase-o-Matic application (example)（Phrase-o-Matic应用（示例）） 70
pillar-boxing（左右加黑边）, video（视频） 354
ping pong Web Workers game (example)（ping pongWeb工作线程游戏（示例）） 484
adding workers（增加工作线程） 491, 492
BE the Browser exercise（扮演浏览器练习） 488, 526
pingPong message handler function for worker（工作线程pingPong消息处理函数） 486

pixels（像素）
accessing in video（视频中访问） 392
in bitmap drawing（位图绘制） 336
drawing on canvas（画布上绘制） 281, 306
as presentation（作为表现）, not content（不是内容） 326
processing in canvas scratch buffer（画布scratch缓冲区中处理） 394, 397
processing video pixels and getting them into canvas for display（处理视频像素并增加到画布上来显示） 396
play button (example)（play按钮（示例））
handler for video booth（摄像间的处理程序） 377
popping back up when video ends（视频结束时弹起） 386
playlist manager（播放列表管理器）, creating（创建） 86
adding code to save playlist（增加代码来保存播放列表） 105
app used to enter song（用来输入歌曲的应用）, click button（点击按钮）, and add song to playlist（向播放列表增加歌曲） 102
code to save the playlist（保存播放列表的代码） 104
displaying playlist on HTML page（HTML页面上显示播放列表） 97
DOM after song titles are added to playlist（歌名增中到播放列表后的DOM） 98, 110
getting song name from text input element（从文本输入元素获取歌名） 94
handling Add Song button click events（处理Add Song点击事件） 89
HTML5 document to hold form and list element for playlist（HTML文档包含表单和播放列表的列表元素） 87
integrating storage code（集成存储代码） 106
playlists（播放列表）
creating video playlist（视频播放列表） 364
implementing for Webville TV (example)（实现Webville TV（示例）） 366
populating with song titles using JavaScript array（使用JavaScript数组填充歌名） 65, 82
play method（play方法）
audio object（audio对象） 533
video object（video对象） 385
plus sign (+)（加号(+)）
addition or string concatenation operator（增加或字符串连接操作符） 45
string concatenation operator（字符串连接操作符） 26
png image format（png图像格式） 347
position object（position对象） 175, 207
coords and timestamp properties（coords和timestamp属性） 190
positionOptions, Geolocation API（地理定位API） 190, 198
postAQuote function (example)（postAQuote函数（示例）） 523
poster attribute（poster属性）, <video> element（<video>元素） 353, 354
poster property（poster属性）, video object（video对象） 406
postMessage method（postMessage方法）
Web Sockets 539
worker object（worker对象） 484, 511, 512
<p> (paragraph) elements（<p>(段落)元素）, changing using JavaScript（使用JavaScript改变） 62
preload attribute（preload属性）, <video> element（<video>元素） 354
presentation（表现）, separate from content（与内容分离）, in canvas（画布中） 326
previewHandler function (example)（previewHandler函数（示例）） 302
calling fillBackgroundColor function（调用fillBackgroundColor函数） 307
updating to call drawText function（更新来调用drawText函数） 330
preview in t-shirt design app（T恤设计应用预览）, problems with（问题） 306
primitive types（基本类型） 40
processFrame function (example)（processFrame函数（示例）） 396
running again（再次运行） 397
processing video frame in canvas scratch buffer（画布scratch缓冲区中处理视频帧） 397
processWork function (example)（processWork函数（示例）） 514, 518
programtheweb.com 271
<progress> element（<progress>元素） 547
progressive video（渐进式视频） 403
properties（属性） 132
accessing（访问）, changing value（修改值）, and enumerating（列举） 133
adding or deleting at any time（任何时间增加或删除） 135
canvas context object（画布上下文对象） 338
fillStyle property（fillStyle属性） 308
text properties（text属性） 328
document object（document对象） 154, 157
element object（element对象） 158
Geolocation API（地理定位API） 190

localStorage, length property（length属性） 424
new（新增）, in CSS3（CSS3中） 548
objects as collections of（对象作为集合） 131
specifying values in JavaScript（JavaScript中指定值） 308
video object（video对象） 363
window object（window对象） 155
pushUnpushButtons helper function（pushUnpushButtons辅助函数） 376, 379, 384
putImageData method（putImageData方法）, canvas context（画布上下文） 397

Q

querySelectorAll method（querySelectorAll方法）, document object（document对象） 376, 542
querySelector method（querySelector方法）, document object（document对象） 542
Quicktime 371
QUOTA_EXCEEDED_ERR exception（QUOTA_EXCEEDED_ERR异常） 458, 468
quotation marks（引号）, double（双引号）, *See* " ", under Symbols（符号）

R

radians（弧度） 316
converting degrees to（度转换为） 317
radio buttons（单选钮） 380, 382
radius parameter of arc method（arc方法的radius参数） 314
reassignWorker function (example)（reassignWorker函数（示例）） 514, 518
rectangles（矩形0, drawing in canvas（画布中绘制） 338
drawing filled rectangle（绘制填充矩形） 292
references（引用）, object（对象） 136
removeItem method（removeItem方法）, localStorage object（localStorage对象） 433, 446
removeStickyFromDOM function (example)（removeStickyFromDOM函数 （示例）） 452
repetitive tasks（重复任务） 46
replaceChild method（replaceChild方法） 270
request/response model based on HTTP（基于HTTP的请求/响应模型） 539
reserved words in JavaScript（JavaScript中的保留字） 41
resizeToWindow function (example)（resizeToWindow函数（示例）） 517
responseText property（responseText属性）, request object（request对象） 222
restore method（restore方法）, canvas context（画布上下文） 540
results from workers' computations（从工作线程计算得到的结果）
from Fractal Explorer workers (example)（ Fractal Explorer 工作线程（示例）） 513
processing in Fractal Explorer (example)（ Fractal Explorer 中处理（示例）） 514
receiving results from workers（从工作线程接收结果） 485, 498
stored in event.data property（存储在event.data属性中） 485
return statements（return语句）
in function body（函数体中） 117
functions without（没有…的函数） 119
RGB color values for pixels（像素的RGB颜色值）, processing video frame data（处理视频帧数据） 397, 410
rotate method（rotate方法）, canvas context（画布上下文） 540

S

Safari 20. 参见 browsers（浏览器）
developer tools for local storage（本地存储的开发工具） 434
H.264 video format（H.264视频格式） 352
HTML5 support（HTML5支持） 18
MP4/H.264 video（MP4/H.264视频） 357
Quicktime player for mp4 video（mp4视频的Quicktime播放器） 371
save and restore methods（保存和恢复方法）, canvas context（画布上下文） 540
Scalable Vector Graphics (SVG)（可缩放矢量图形（SVG）） 537
sci-fi effect for video（视频的科幻效果） 374, 400, 410
scope（作用域）, variables（变量） 124
scratch buffer（scratch缓冲区）, video processing with（视频处理） 390, 393
implementing buffer with canvas（用画布实现缓冲区） 395~398
<script> elements（<script>元素） 27
adding to HTML file for call to Twitter JSONP API（增加到HTML文件来调用Twitter JSONP API） 322
adding to HTML in <head> or <body>（增加到HTML的<head> 或<body>中） 53

creating and inserting dynamically（动态地创建和插入） 263, 267~269
retrieving data with（用来获取数据） 248~251, 257
script injection（脚本注入） 271
specifying in HTML5（HTML5中指定） 5
scrollMapToPosition function (example)（scrollMapToPosition函数（示例）） 204
adding to application（增加到应用） 205
<section> element（<section>元素） 546
security（安全性）, JSONP and 259
security policy（安全策略）, browsers（浏览器） 244
selectedIndex property（selectedIndex属性）, selection form controls（选择表单控件） 302
how it works（如何工作） 303
<select> element（<select>元素） 301, 453
Selectors API（选择器API） 542
selectors（选择器）, new（新增）, in CSS3（CSS3中） 548
semicolon (;)（分号（;））, ending JavaScript statements（结束JavaScript语句） 39
send method（send方法）, XMLHttpRequest object（XMLHttpRequest对象） 221
servers（服务器） 230
moving to live server（转移到实际服务器） 237
problem when moving to live server（转移到实际服务器的问题） 242
setting up your own Web server（建立你自己的Web服务器） 231
sessionStorage object（sessionStorage对象） 460
setAttribute method（setAttribute方法）, element object（element对象） 158, 274, 275
setting sticky's id to its unique key（将即时贴的id设置为它的唯一键） 450
using to set the class attribute（用来设置class属性） 236, 257, 379, 430
using to set the id attribute（用来设置id属性） 267, 269, 450
using to set the src attribute（用来设置src属性） 267, 269
setEffect handler function（setEffect处理函数）, video booth (example)（摄像间（示例）） 378, 391
setInterval method（setInterval方法）, window object（window对象） 263, 265
using with Web Workers（结合Web工作线程使用） 523
setItem method（setItem方法）, localStorage object（localStorage对象） 418, 421
setTimeout method（setTimeout方法）, window object（window对象）
timeout parameter of 0（timeout参数为0） 398
using to process video frame data（用来处理视频帧数据） 397
using with Web Workers（结合Web工作线程使用） 523
setupGraphics function (example)（setupGraphics函数（示例）） 509
setVideo handler function（setVideo处理函数）, video booth (example)（摄像间（示例）） 378, 387
SGML 9
shadowBlur property（shadowBlur属性）, canvas context（画布上下文） 335
shadowColor property（shadowColor属性）, canvas context（画布上下文） 335
shadowing variables（遮蔽变量） 126
shadowOffsetX and shadowOffsetY properties（shadowOffsetX和shadowOffsetY属性）, canvas context（画布上下文） 335
showMap function (example)（showMap函数（示例）） 184
creating map and displaying marker for initial location（创建地图并显示初始位置标记） 205
making sure it's called only once（确保只调用一次） 195
single-threaded model（单线程模型）, JavaScript 474, 477
breaking down（分解） 475
slashes (//)（斜线(//)）, beginning JavaScript comments（开始JavaScript注释） 39
slow script message（脚本运行缓慢消息） 473
<source> element（<source>元素）
src attribute（src属性） 359
type attribute（type属性） 359
using inside <video> element for each video format（为各视频格式在<video>元素中使用） 358
special effects（特效）
applying to videos（应用到视频） 389~391
functions（函数） 399, 410
speed property（speed属性）, coordinates object（coordinates对象） 190, 197
splice method（splice方法）, Array object（Array对象） 449
SQL, Web 543
square brackets ([ ])（中括号（[ ]））
accessing object properties（访问对象属性） 133
and associative arrays（关联数组） 424
creating and indexing arrays（创建和索引数组） 67
using with localStorage（用于localStorage） 424
squares（方块）, drawing on canvas（画布上绘制） 302
creating with fillRect（用fillRect创建） 290, 292, 304
filling background color of canvas before drawing new squares（绘制新方块之前填充画布背景色） 306

pseudo-code for drawSquare function（drawSquare函数的伪代码） 303
random x, y, and width of squares（方块的随机x,y和宽度） 308
writing drawSquare function（编写drawSquare函数） 304
src attribute（src属性）
<script> element（<script>元素） 53, 218, 249
updating with setAttribute（用setAttribute属性更新） 267
<source> element（<source>元素） 358, 359
<video> element（<video>元素） 353, 354
src property（src属性）
audio object（audio对象） 533
image object（image对象） 333
video object（video对象） 370
startWorkers function (example)（startWorkers函数（示例）） 509, 510
statements（语句） 37
ending with semicolon（用分号结束） 39
stickies application (example)（即时贴应用（示例）） 418, 428
adding "Add Sticky Note to Self" button（增加"Add Sticky Note to Self"按钮） 431
adding JavaScript code（增加JavaScript代码） 430
converting createSticky to use an array（转换createSticky来使用数组） 441
creating interface（创建界面） 429
deleting sticky from DOM（从DOM删除即时贴） 452
deleting sticky notes（删除便条贴） 446
design flaw（设计缺陷） 436
integrating array-based code（集成基于数组的代码） 443
rewriting to use an array（重写为使用数组） 440
selecting sticky note to delete（选择要删除的即时贴） 450
updating user interface to specify color（更新用户界面来指定颜色） 453~456
streaming video（流式视频） 403
technologies for（技术） 404
string concatenation operator (+)（字符串连接操作符(+)） 26, 45
string expressions（字符串表达式） 43
stringify method（stringify方法）. *See* JSON.stringify method（JSON.stringify方法）
strings（字符串）
accessing and enumerating object properties（访问和列举对象属性） 133
in arrays（数组中） 71
as associative array indexes（作为关联数组索引） 424
conversions to numbers in expressions（表达式中转换为数字） 45
converting objects to JSON string format（对象转换为JSON字符串格式） 226
converting to floats with parseFloat function（用parseFloat函数转换为浮点数） 423
converting to integers with parseInt function（用parseInt函数转换为整数） 423
creating string representation of an array（创建数组的字符串表示） 441, 467
key/value pairs stored in local storage（本地存储中存储的键/值对） 418
as objects（作为对象） 159
primitive type in JavaScript（JavaScript中的基本类型） 40
receiving from Web Workers with onmessage in event.data property（用event.data属性中onmessage接收Web工作线程） 485
sending to Web Workers with postMessage（用postMessage发送到Web工作线程） 484
stroke method（stroke方法）, canvas context（画布上下文） 312
strokeText method（strokeText方法）, canvas context（画布上下文） 328
structure（结构） 35, 545
<style> element（<style>元素）
adding border to canvas（向画布增加边框） 288
CSS is style standard（CSS是样式标准） 9, 31
style property（style属性） 455
subworkers（子工作线程） 523
success handler（成功处理程序）, Geolocation API（地理定位API） 174, 175, 190
SVG (Scalable Vector Graphics)（SVG（可缩放矢量图形）） 537

T

table and table-cell layouts（表与表单元格布局） 548
target property（target属性）, event object（event对象） 451, 485
task monitor on OS X or Windows（OS X或Windows上的任务监视器） 520
tasks（任务）, sending and receiving data from Web Workers (Fractal Explorer example)（从Web工作线程发送和接收数据（Fractal Explorer示例）） 512
terminate method（terminate方法）, worker object（worker对象） 522

www.pin51.com 最新最全 上海科学技术文献出版社

textAlign property（textAlign属性）, canvas context（画布上下文） 328
aligning tweet text in t-shirt design app (example)（T恤设计应用中对齐微博文本（示例）） 331
textBaseline property（textBaseline属性）, canvas context（画布上下文） 329
text（文本）, drawing on canvas（画布上绘制） 325~332, 338
displaying HTML entities（显示HTML实体） 335
drawText function（drawText函数） 345
splitting it into lines（划分为多行） 335
text methods and properties in canvas API（画布API中的文本方法和属性）328
text <input> element（文本<input>元素）, value property（value属性） 94
checking whether user entered input（检查用户是否已输入） 96
Theora video format（Theora视频格式） 357
third-party hosting services（第三方托管服务） 230, 232
this (keyword)（this（关键字）） 144
adding to movie object (example)（增加到movie对象（示例）） 145
questions and answers about（问答） 151
using with constructors（用于构造函数） 147
using with method calls（用于方法调用） 149, 151
threading（线程）. 参见 Web Workers（Web工作线程）
adding another thread of control（另一个控制线程） 476
single-threaded model（单线程模型）, JavaScript 474
with Web Workers（用Web工作线程） 478, 524
time（时间）
Date object（Date对象）, getTime method（getTime方法） 442
milliseconds since 1970（自1970年以来的毫秒数） 442
<time> element（<time>元素） 547
timeout option （timeout选项） 199, 201
timestamp property（timestamp属性）, position object（position对象） 190
timeupdate event（timeupdate事件） 398
title property（title属性）, document object（document对象） 157
toDataURL method（toDataURL方法）, canvas object（canvas对象） 347
toggle buttons（开关按钮） 380, 382
tracking movements（跟踪移动） 192~198
form to start and stop tracking（开始和停止跟踪的表单） 193
transition property（transition属性） 548
translate and rotate methods（转换和旋转方法）,canvas context（画布上下文）540
triangles（三角形）, drawing on canvas（画布上绘制） 311
true and false (boolean values)（true和false（布尔值）） 39
try/catch statements（try/catch语句）,capturing exceptions（捕获异常）458, 468
t-shirt（T恤） 282. 参见 TweetShirt Web Application
TweetShirt Web Application (example)（TweetShirt Web 应用（示例）） 282
adding tweets to <select> element in <form> （向<form>中<select>元素增加微博） 323
adding user interface form to HTML page（向HTML页面增加用户界面表单） 301
creating application design（创建应用设计） 297
drawing an image（绘制图像） 333
drawing circles（画圆） 318
drawing squares（画方块） 304
drawing text（绘制文本） 324, 327, 330, 331
filling the background color（填充背景色） 306
form for application interface（应用界面表单） 298
getting tweets from Twitter（从Twitter获取微博） 322
making image of design to upload and have printed on shirt（建立设计图像上传并打印在T恤上） 347
requirements and user interface（需求和用户界面） 283
reviewing implementation plan（检查实现计划） 296
Twitter JSONP API, making call to（调用） 322
type attribute（type属性）
removal from <link> and <script> tags（从<link><script>标记删除） 5
<source> element（<source>元素） 359

U

ul.appendChild method（ul.appendChild方法） 101
undefined values（定义值） 73
returned by functions without return statement（无return语句的函数返回） 121
underscore (_)（下划线(_)）, beginning variable names（开始变量名） 40, 42
updateSales function (example)（updateSales函数（示例）） 230
updateTweets callback function (example)（updateTweets函数（示例）） 323
URL property（URL属性）, document object（document对象） 157
URLs
callback parameter（回调参数） 254

setting up JSONP URL (example)（建立JSONP URL（示例））267
updating JSON URL to include last report time (example)（更新JSONP URL来包含最后一次报告时间（示例））275
Web Socket 539
work around for browser caching（避开浏览器缓存） 272
UTF-8 9, 31

V

value attribute（value属性）, text <input> element（文本<input>元素） 95
value property（value属性）, text <input> element（文本<input>元素） 94
value attribute versus（value属性与） 95
values（值）
changing object property values（修改对象属性值） 133
functions as（函数作为） 128, 129
object property values（对象属性值） 132
variables（变量）
assigning functions to（指定函数） 128
chaining value of（串链值） 39, 133, 141
comparing to empty string（与空串比较） 108
declaring and assigning value（声明和赋值） 26, 38
local and global（局部和全局） 123, 160
naming（命名） 40, 42
objects assigned to（对象赋为） 136
passing to functions（传递到函数） 121
scope of（作用域） 124
shadowing（遮蔽） 126
short life of（短暂一生） 125
var keyword（var关键字） 39
vector fonts（向量字体） 329
vector graphics vs. bitmap（向量图形与位图） 336
video（视频） 16, 349~412
adding format information in the <source> element（<source>元素中增加格式信息） 359
booth (example)（摄像间（示例）） 373
adding special effects（增加特效） 389~391
code to process the video（处理视频的代码） 396
demo unit（演示单元） 374~376
getting demo videos ready（获取完成的演示视频） 383
helper functions（辅助函数） 379
implementing video controls（实现视频控件） 384~386
overview of video processing（视频处理概览） 392
setEffect and setVideo handlers（setEffect和setVideo处理程序） 378
switching test videos（切换测试视频） 387
video processing using scratch buffer（使用scratch缓冲区处理视频） 393
writing special effects（编写特效） 399~404
browser support（浏览器支持）, determining level of（确定级别） 361, 411
canPlayType method（canPlayType方法）, how it works（如何工作） 369~375
codecs 357, 358
controls' appearance in different browsers（不同浏览器中的控件外观） 355
Crossword Puzzle（填字游戏） 409, 411
error event（error事件）, using（使用） 406
errors（错误） 405
falling back to supported player（后备的支持播放器） 362
formats（格式） 352, 356, 358
and possibility of standardization（标准的可能性） 360
hosted on the web（在web上托管） 403
how <video> element works（<video>元素如何工作） 353
ideas for further development（进一步开发的想法） 407
No Dumb Questions（没有傻问题） 360, 371
Sharpen Your Pencil exercise（动动笔练习）
control buttons（控制按钮）, toggle or radio（开关或单选） 380, 382
implementing playlist（实现播放列表） 364, 365
western and sci-fi effects（西部片和科幻片效果） 400, 410
streaming（流式） 403
summary of important points（要点总结） 408
testing for browser support when using code to load video（使用代码加载视频时测试浏览器支持） 368
things to watch out for（当心的问题） 371
using JavaScript with HTML5（结合HTML5使用JavaScript） 23
Webville TV (example)（Webville TV（示例）） 350
building with HTML5 technology（用HTML5技术构建） 350
designing the playlist（设计播放列表） 365
handler for ended event to go to next video（ended事件的处理程序来播放下一个视频） 367
HTML5 page（HTML5页面） 351
implementing getFormatExtension function with canPlayType（结合canPlayType使用getFormatExtension实现） 369
implementing nextVideo function（实现nextVideo函数） 367

www.aibbt.com 让未来触手可及

implementing video playlist（实现视频播放列表） 366
integrating getFormatExtension function（集成getFormatExtension函数） 370
<video> element（<video>元素）
attributes（属性） 354, 408
how it works（如何工作） 353
interview with（访谈） 388
methods（方法）, properties（属性）, and events（事件） 363
new HTML5 element and API（新增HTML5元素和API） 351
<object> element within（其中的<object>元素） 362
partnership with <canvas> element（与<canvas>元素的伙伴关系） 339, 388
<source> element within（其中的<source>元素） 358
video object（video对象）
accessing frame data（访问帧数据） 396
canPlayType method（canPlayType方法） 368~374
error property（error属性） 406
load method（load方法） 385
loop property（loop属性） 385
methods（方法）, properties（属性）, and events（事件） 363
muted property（muted属性） 385
pause method（pause方法） 385
play method（方法） 366, 385
properties（属性）, methods（方法）, and events（事件） 408
src property（src属性） 366, 387
volume property（volume属性） 360
viewport（视窗） 354
volume property（volume属性）
audio object（audio对象） 533
video object（video对象） 360
Vorbis audio codec（Vorbis音频编解码器） 357
VP8 video codec（VP8视频编解码器） 357

W

W3C 20
WampServer 231
watchId variable (example)（watchId变量（示例）） 194
Watch it!（当心！）
browsers not supporting XMLHttpRequest's onload property（不支持XMLHttpRequest onload属性的浏览器） 229
deleting all items from local store（从本地存储删除所有数据项） 435
ensuring that server is serving video files of correct MIME type（确保服务器提供正确MIME类型的视频文件） 371
grabbing image from canvas（从画布获取图像）, and code run from file://（从file://运行的代码） 347
Internet Explorer not supporting Web Workers prior to IE10（IE10以前版本的Internet Explore不支持Web工作线程） 482
local storage and browser issues with file://（本地存储和file://的浏览器问题） 422
pages served from file://（从file://提供页面）, security restrictions in Chrome（Chrome中的安全限制） 371
pushing browser over local storage limit（让浏览器超出本地存储限制） 459
Quicktime need to play mp4 video in Safari（Safari中需要Quicktime播放mp4视频） 371
rapid changes in video support by browsers（浏览器对视频的支持快速变化） 411
Ready Bake Code not working in some browsers（一些浏览器中不能工作的成品代码） 104
security restrictions in Chrome preventing running of Web Workers from file（Chrome不允许从文件运行Web工作线程的安全限制） 482
server required to test geolocation code on mobile devices（移动设备上测试地理定位代码所需的服务器） 179
work around for browsers not supporting XMLHttpRequest's onload property（避开不支持XMLHttpRequest onload属性的浏览器） 241
watchLocation function (example)（watchLocation函数（示例）） 194
watchPosition method（watchPosition方法）, geolocation object（geolocation对象） 190, 192, 194, 207
controlling updates of location（控制位置更新） 197
too many calls to displayLocation（太多displayLocation调用） 206
wav audio format（wav音频格式） 533
web applications（web应用）
APIs to create（创建…的API） 15
examples（示例） 22
and HTML5 6, 13
and JavaScript 21, 24
offline（离线） 538
what is it（是什么） 28
WebKit-based browsers（基于WebKit的浏览器） 20 参见 browsers （浏览器）
HTML5 support（HTML5支持） 18

www.minxue51.com 最新资料

.webm video file format（.webm视频文件格式） 352, 357
WebM/VP8 video container format（WebM/VP8视频容器格式） 357
Web pages vs. Web applications（Web页面与Web应用） 28
Web Platform Installer (Microsoft)（Web平台安装工具(Microsoft)） 231
web services（web服务） 213
how it works（如何工作）, Mighty Gumball (example)（Mighty Gumball（示例）） 216
JSONP security issues and（JSONP安全问题） 259
keeping an open connection with（保持打开连接）, using Web Sockets（使用Web Sockets） 539
lucky/unlucky service（幸运/不幸运服务） 224
parameters supported（支持的参数） 271, 274
receiving JSON data from（从中接收JSON数据） 233
specifying callback function for（指定回调函数） 254
using JSONP with（使用JSONP） 253
using XMLHttpRequest with（使用XMLHttpRequest） 220
ways to access（访问途径）, in public API（公共API中） 271
XMLHttpRequest, cross domain security issues with（跨域安全问题） 244
Web Sharing (Mac)（Web共享（Mac）） 231
Web Sockets 539
Web SQL 543
Web Storage API 108, 418. 参见 local storage（本地存储）; localStorage object（localStorage对象）
support for（支持） 422
Web Workers（Web工作线程） 16, 473~530
adding another thread of control（另一个控制线程） 476
adding workers to pingPong game (example)（向pingPong游戏增加工作线程（示例）） 491, 492
BE the Browser exercise（扮演浏览器练习） 488, 526
browsers' support of（浏览器支持） 482
building explorer for Mandelbrot Set（为Mandelbrot集构建浏览器） 494
building Fractal Explorer application (example)（构建Fractal Explorer应用（示例）） 503, 509
creating（创建） 483
creating and giving tasks to（创建和分配任务） 508
Crossword Puzzle（填字游戏） 525, 528
data types that can be sent to workers（可以发送到工作线程的数据类型） 484
getting workers started in Fractal Explorer application（Fractal Explorer应用中启动工作线程） 510
handling click event to zoom in on canvas in Fractal Explorer (example)（Fractal Explorer中处理点击事件在画布上放大（示例）） 515
handling errors in workers（工作线程中处理错误） 522
how they work（如何工作） 478
how workers make your apps faster（工作线程如何使用应用更快） 500
implementing in Fractal Explorer application（Fractal Explorer应用中实现） 511
importScripts global function（importScripts全局函数） 493
managing fractal generations in Fractal Viewer（Fractal Viewer中管理分形代） 518
No Dumb Questions（没有傻问题） 491
number of workers（工作线程个数）
effects on performance（对性能的影响） 520
limits on（限制） 501
processing workers' results in Fractal Explorer（Fractal Explorer中处理工作线程的结果） 514
ready-baked code for workers' computation of Mandelbrot Set（工作线程计算Mandelbrot集的成品代码） 504~507
receiving a message from the worker（从工作线程接收消息） 485
results from workers' computations（工作线程计算的结果） 513
rewriting pseudo-code to use workers（重写伪代码来使用工作线程） 502
sending a message to the worker（向工作线程发送消息） 484
Sharpen Your Pencil exercise（动动笔练习）
potential uses for workers（工作线程的潜在使用） 481
using compact workers（使用简洁的工作线程） 490, 527
subworkers（子工作线程） 523
summary of important points（要点总结） 524
tasks for Fractal Explorer workers（Fractal Explorer工作线程的任务） 512
terminating（终止） 522
using importScripts to make JSONP requests（使用importScripts建立JSONP请求） 523
using multiple workers to compute Mandelbrot Set（使用多个工作线程来计算Mandelbrot集） 497~500
why workers can't access the DOM（为什么工作线程不能访问DOM） 480
writing worker's message handler（编写工作线程的消息处理程序） 486
western effect for video（视频的西部片效果） 374, 400, 410
"Wherever you go, there you are" (Geolocation example)（"无处可逃"（地理定位示例）） 192

pin51.com 拼吾爱 最新最全 下载站 资源分享

while loops（while循环） 46
- deciding between for loops and（选择for循环还是…） 47
- evaluating (example)（执行（示例）） 48
- example in JavaScript（JavaScript中的示例） 26
- if/else statements in（if/else语句） 50

white space in JavaScript code（JavaScript代码中的空白符） 39

width and height attributes（width和height属性）
- <canvas> element（<canvas>元素） 286
  - setting using CSS（用CSS设置） 289
- <video> element（<video>元素） 353, 354

WiFi positioning（WiFi定位） 169

window object（window对象） 154
- creating onload event handler for（创建onload事件处理程序） 64, 75, 159
- document object property（document对象属性） 155
- as global object（作为全局对象） 156, 158
- localStorage property（localStorage属性） 422
- location property（location属性） 347
- onload property（onload属性） 64, 156
- properties and methods（属性和方法） 155
- setInterval method（setInterval方法） 155, 265
- setTimeout method（setTimeout方法） 155, 397

windows（窗口）
- adding info window for Google Maps marker (example)（为Google Maps标记增加信息窗口（示例）） 187
- fitting canvas to browser window in Fractal Viewer (example)（Fractal Viewer中让画布占满浏览器窗口（示例）） 517

Windows systems（Windows系统）
- installing Web Server on（安装Web服务器） 231
- making sure server is serving video with correct MIME type（确保服务器提供正确MIME类型的视频） 371
- task monitor（任务监视器） 520

worker object（worker对象） 参见 Web Workers（Web工作线程）
- close method（close方法） 522
- creating（创建） 483
- creating and using multiple（创建和使用多个） 491, 492
- onerror property（onerror属性） 522
- onmessage property（onmessage属性） 485
- postMessage method（postMessage方法） 484
- subworkers（子工作线程） 523
- terminate method（terminate方法） 522

X

XHTML 9, 536
- problems with（问题） 11

XML
- JSON versus（JSON与） 226, 271
- SVG graphics（SVG图形） 537
- uses of XHTML for（使用XHTML） 536

XMLHttpRequest object（XMLHttpRequest对象） 220, 239
- accessing response text（访问响应文本） 222
- cross-domain requests（跨域请求）, security issues with（安全问题） 244, 277
- Crossword Puzzle（填字游戏） 278, 280
- Fireside Chat with JSONP （与JSONP的围炉夜话） 260
- interview with（访谈） 225, 240
- Level 2 240
- onload handler function（onload处理函数） 229
- requests made by workers（工作线程的请求） 491
- retrieving JSONP data with（用来获取JSONP数据） 233
- server required for use of（使用…需要服务器） 230
- when to use（何时使用） 246, 277
- work around for browsers not supporting Level 2（避开不支持Level 2的浏览器） 241

Z

zooming in on canvas in Fractal Viewer (example)（Fractal Viewer中在画布上放大（示例）） 515

pin51.com
拼吾爱
最新[illegible]资源分享下载站

最新最全资源分享 拼吾爱 pin5i.com

版本记录



所有内部版面设计都由Eric Freeman和Elisabeth Robson完成。

Kathy Sierra和Bert Bates始创了Head First系列的外观。

版面设计和制版都完全在Apple Macintosh上完成准确地讲是两台Mac Pro和两台MacBook Air。

写作地点包括：Bainbridge Island, Washington; Portland, Oregon; Las Vegas, Nevada; Port of Ness, Scotland; Seaside, Florida; Lexington, Kentucky; Tucson, Arizona以及Anaheim, California。漫长的写书过程中，我们用Honest Tea和GT's Kombucha来提神，另外离不开Sia、Sigur Ros、Tom Waits、OMD、Phillip Glass、Muse、Eno、Krishna Das、Mike Oldfield、Audra Mae、Devo、Steve Roach、Beyman Brothers和Pogo（都出自turntable.fm）的音乐，还有很多你可能不太关心的20世纪80年代的音乐。

pin5i.com 拼吾爱 最新最全资源分享下载站

不是说再见

潜心加入我们的

wickedlysmart.com

